U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

José Luís da Silva Devezas

# Graph-Based Entity-Oriented Search

Supervisor: Sérgio Nunes (U.Porto)

January 2021

Universidade do Minho

universidade de aveiro

MAPi DOCTORAL PROGRAMME IN COMPUTER SCIENCE

DOCTORAL PROGRAM IN COMPUTER SCIENCE OF THE UNIVERSITIES OF MINHO, AVEIRO AND PORTO

*Thesis submitted to Faculty of Engineering of the University of Porto for the Doctor Degree in Computer Science within the Joint Doctoral Program in Computer Science of the Universities of Minho, Aveiro and Porto.*

U.PORTO INESC TEC and Faculty of Engineering, University of Porto

**Jury president:**

- Doctor Gabriel de Sousa Torcato David, Associate Professor, Department of Informatics Engineering, Faculty of Engineering, University of Porto;

**Jury members:**

- Doctor Krisztian Balog, Full Professor, Department of Electrical Engineering and Computer Science, University of Stavanger, Norway;
- Doctor Bruno Emanuel da Graça Martins, Assistant Professor, Department of Computer Science and Engineering, Instituto Superior Técnico, University of Lisbon;
- Doctor António Joaquim da Silva Teixeira, Associate Professor with Aggregation, Department of Electronics, Telecommunications and Informatics, University of Aveiro;
- Doctor Pedro Manuel Pinto Ribeiro, Assistant Professor, Department of Computer Science, Faculty of Sciences, University of Porto;
- Doctor João Manuel Paiva Cardoso, Full Professor, Department of Informatics Engineering, Faculty of Engineering, University of Porto;
- Doctor Sérgio Sobral Nunes, Assistant Professor, Department of Informatics Engineering, Faculty of Engineering, University of Porto (Supervisor).

José Luís da Silva Devezas: *Graph-Based Entity-Oriented Search,* Doctoral Thesis, Doctoral Program in Computer Science of the Universities of Minho, Aveiro and Porto, January 2021.

WEBSITE:
http://www.josedevezas.com

E-MAIL:
joseluisdevezas@gmail.com

# ABSTRACT

Entity-oriented search has revolutionized search engines. In the era of Google Knowledge Graph and Microsoft Satori, users demand an effortless process of search. Whether they express an information need through a keyword query, expecting documents and entities, or through a clicked entity, expecting related entities, there is an inherent need for the combination of corpora and knowledge bases to obtain an answer. Such integration frequently relies on independent signals extracted from inverted indexes, and from quad indexes indirectly accessed through queries to a triplestore. However, relying on two separate representation models inhibits the effective cross-referencing of information, discarding otherwise available relations that could lead to a better ranking. Moreover, different retrieval tasks often demand separate implementations, although the problem is, at its core, the same. With the goal of harnessing all available information to optimize retrieval, we explore joint representation models of documents and entities, while taking a step towards the definition of a more general retrieval approach. Specifically, we propose that graphs should be used to incorporate explicit and implicit information derived from the relations between text found in corpora and entities found in knowledge bases. We also take advantage of this framework to elaborate a general model for entity-oriented search, proposing a universal ranking function for the tasks of ad hoc document retrieval (leveraging entities), ad hoc entity retrieval, and entity list completion.

At a conceptual stage, we begin by proposing the graph-of-entity, based on the relations between combinations of term and entity nodes. We introduce the entity weight as the corresponding ranking function, relying on the idea of seed nodes for representing the query, either directly through term nodes, or based on the expansion to adjacent entity nodes. The score is computed based on a series of geodesic distances to the remaining nodes, providing a ranking for the documents (or entities) in the graph. In order to improve on the low scalability of the graph-of-entity, we then redesigned this model in a way that reduced the number of edges in relation to the number of nodes, by relying on the hypergraph data structure. The resulting model, which we called hypergraph-of-entity, is the main contribution of this thesis. The obtained reduction was achieved by replacing binary edges with $n$-ary relations based on sets of nodes and entities (undirected document hyperedges), sets of entities (undirected hyperedges, either based on co-occurrence or a grouping by semantic subject), and pairs of a set of terms and a set of one entity (directed hyperedges, mapping text to an object).

We introduce the random walk score as the corresponding ranking function, relying on the same idea of seed nodes, similar to the entity weight in the graph-of-entity. Scoring based on this function is highly reliant on the structure of the hypergraph, which we call representation-driven retrieval. As such, we explore several extensions of the hypergraph-of-entity, including relations of synonymy, or contextual similarity, as well as different weighting functions per node and hyperedge type. We also propose TF-bins as a discretization for representing term frequency in the hypergraph-of-entity. For the random walk score, we propose and explore several parameters, including length and repeats, with or without seed node expansion, direction, or weights, and with or without a certain degree of node and/or hyperedge fatigue, a concept that we also propose.

For evaluation, we took advantage of TREC 2017 OpenSearch track, which relied on an online evaluation process based on the Living Labs API, and we also participated in TREC 2018 Common Core track, which was based on the newly introduced TREC Washington Post Corpus. Our main experiments were supported on the INEX 2009 Wikipedia collection, which proved to be a fundamental test collection for assessing retrieval effectiveness across multiple tasks. At first, our experiments solely focused on ad hoc document retrieval, ensuring that the model performed adequately for a classical task. We then expanded the work to cover all three entity-oriented search tasks. Results supported the viability of a general retrieval model, opening novel challenges in information retrieval, and proposing a new path towards generality in this area.

# RESUMO

A pesquisa orientada a entidades revolucionou os motores de busca. Na era do Google Knowledge Graph e do Microsoft Satori, o utilizador exige um processo de pesquisa sem esforço. Quer o utilizador expresse uma necessidade de informação através de uma consulta de palavras-chave, procurando obter documentos e entidades, ou através de uma entidade clicada, procurando obter entidades, existe a necessidade inerente da combinação de corpora e bases de conhecimento para obter uma resposta. Tal integração depende frequentemente de sinais independentes extraídos de índices invertidos, e de índices de quads acedidos indiretamente através de interrogações a uma triplestore. Contudo, depender de dois modelos de representação separados restringe a eficácia do cruzamento de informação, descartando relações que, de outra forma, estariam disponíveis e poderiam resultar num ranking de maior qualidade. Além disso, diferentes tarefas de recuperação exigem frequentemente implementações separadas, apesar do problema ser, na prática, o mesmo. Com o objetivo de tirar partido de toda a informação disponível para otimizar a recuperação, exploramos modelos de representação conjunta de documentos e entidades, em simultâneo dando um passo em direção à definição de uma abordagem mais geral para a recuperação de informação. Concretamente, propomos a utilização de grafos para incorporar informação explícita e implícita derivada das relações entre o texto que encontramos em corpora e as entidades que encontramos em bases de conhecimento. Tiramos também partido deste framework para elaborar um modelo geral para pesquisa orientada a entidades, propondo uma função universal de ranking para as tarefas de ad hoc document retrieval (alavancando entidades), ad hoc entity retrieval, e entity list completion.

Na fase conceptual, começamos por propor o graph-of-entity, baseado nas relações entre diferentes combinações de nós termo e entidade. Introduzimos o entity weight como a função de ranking correspondente, suportada na ideia de nós-semente que representam a consulta, quer diretamente através de nós termo, quer baseados na expansão da consulta a entidades adjacentes. A pontuação é computada com base numa série de distâncias geodésicas aos restantes nós, resultando num ranking de documentos (ou entidades) no grafo. Com a finalidade de melhorar a baixa escalabilidade do graph-of-entity, redesenhamos este modelo de forma a reduzir o número de arestas em relação ao número de nós, suportando-nos no hipergrafo como estrutura de dados. O modelo resultante, ao qual chamamos hypergraph-of-entity, é a principal contribuição desta tese. A redução obtida foi alcançada substituindo arestas binárias por relações $n$-árias baseadas em conjuntos de nós e entidades (hiperaresta não-dirigida documento), conjuntos de entidades (hiperarestas não-dirigidas, baseadas em co-ocorrência ou no agrupamento por sujeito semântico), e pares de um conjunto de termos e um conjunto de uma entidade (hiperarestas dirigidas, mapeando texto para um objeto).

Introduzimos a random walk score como a função de ranking correspondente, suportada na mesma ideia de nós-semente utilizada também no entity weight do graph-of-entity. A pontuação gerada por esta função é altamente dependente da estrutura do hipergrafo, o que designamos por recuperação orientada à representação. Como tal, exploramos várias extensões do hypergraph-of-entity, incluindo relações de sinonímia, ou similaridade contextual, bem como diferentes funções de pesagem por tipo de nó e hiperaresta. Também propomos o conceito de TF-bins como uma discretização para representar a frequência do termo no hipergrafo. Para a random walk score, propomos e exploramos vários parâmetros, incluindo comprimento e repetições, com ou sem a expansão de nós-semente, direção, ou pesos, e com ou sem um certo grau de fadiga nos nós e/ou hiperarestas, um conceito também por nós proposto.

Para a avaliação, tiramos partido do TREC 2017 OpenSearch, suportado num processo de avaliação online com base na API do Living Labs, e também participamos no TREC 2018 Common Core, baseado no recém-introduzido TREC Washington Post Corpus. As nossas principais experiências suportaram-se na coleção da Wikipedia do INEX 2009, que veio a demonstrar-se fundamental ao teste e à avaliação da eficácia da recuperação, de forma transversal às múltiplas tarefas. Inicialmente, as nossas experiências focaram-se exclusivamente em ad hoc document retrieval, garantindo que o modelo desempenhava adequadamente esta tarefa clássica. De seguida, expandimos o trabalho de forma a englobar as três tarefas de pesquisa orientada a entidades. Os resultados obtidos suportam a viabilidade de um modelo de recuperação geral, abrindo novos desafios na recuperação de informação e propondo um novo caminho para a generalidade nesta área.

# ACKNOWLEDGEMENTS

I thank my colleagues, professors and friends at FEUP InfoLab for all the support I have received throughout this journey, and for their patience in dealing with my complexities. It has been a long hard road, both personally and professionally, and we have shared a lot of the load together. In particular, I thank Yulia Karimova for her constant friendship and motivation, João Castro for our general discussions on life-as-a-PhD and on methodological approaches, particularly regarding literature review, Inês Koch and Joana Rodrigues for their kindness and for listening to my multiple rambles about hypergraphs and life in general, João Rocha for helping me throughout technical, scientific and personal challenges in his own amazing way, João Correia Lopes for his constant good advice and the promotion of "tunnel vision", for sharing a good book on graph data management, and for his overall friendship and interesting moments of storytelling, good laughs and musical recommendations, Carla Teixeira Lopes for her brief but frequent interventions with motivating words about my research and my journey through teaching, as well as for the good literature recommendations and our discussions on the graph-of-entity, and Cristina Ribeiro for all the hard work she puts in for FEUP InfoLab, for suggestions on fundamental information retrieval literature, and for her guidance, along with that of Maria Eugénia Fernandes, who I also thank, during my work with DSpace Auditor. I thank Antonio Espejo, from the Department of Information Systems and Languages of the University of Alicante, for our collaboration in TREC 2018, during his stay at FEUP InfoLab, where I got the idea to use keyword-based document profiles to reduce collection size. I also thank Melinda Oroszlányová and Nelson Pereira, who were a pleasure to work with, and helped me through times of self-doubt.

I thank all of the ANT team and, in particular, Tiago Devezas, who is not only a friend, but also a master front-end builder, without whose work ANT would reach no one. I thank Noémia Moreira for her Master's thesis, where she directly contributed to improving ANT's user experience. I thank Ricardo Amorim for unofficially dedicating some of his free time to contributing to ANT's Android app, and for, later on, taking the time to read my rambles about life. I also thank all of the students I co-supervised, Inês Garganta, André Pires, and João Damas, who taught me as much, if not more, as I did them.

I thank MAP-i for the multiple opportunities provided to their students. In particular, there have been numerous discussions with colleagues and professors that have helped question, position and motivate this work. Having access to three of the best universities in Portugal was an amazing opportunity! In particular, I thank my MAP-i colleagues, Ricardo Cruz for his friendship and support, enthusiastic suggestions and overall availability, and Gil Coutinho for his expeditious help with server infrastructure at INESC TEC, as well as professors António Teixeira, from the University of Aveiro, for introducing me to information extraction topics like relation extraction, and for our discussions at conferences over multiple scientific and organizational topics, and José João Almeida, from the University of Minho, for his Donald Duck example that I have been using to distinguish between taxonomies and ontologies, and for taking the time to discuss some of my thesis ideas and providing feedback as we briefly met during events. I would also like to thank all the MAP-i directors and, in particular, Jorge Sousa Pinto for his motivating words during our discussion on my third year progress report, and Gabriel David for his continued support, both in finding quick solutions to general issues and in helping to manage some of the bureaucratic stresses that come with the program. These have also been mitigated by the hard work of Sofia Santos, at the central office for the program, in the Faculty of Sciences of the University of Porto, and by Conceição Barbosa, from the Department of Informatics of the University of Minho, during the initial stages of registration.

I thank my good friends, Filipe Coelho and Gustavo Laboreiro, for our decade-old conversations on information retrieval and machine learning, over so many lunches and coffee breaks, both at and outside FEUP. I thank Carlos Tavares for making me think about quantum and for our summer discussions on so many scientific, metaphysics and overall interesting topics. I thank my good old friend, Nuno Cravino, for his suggestion on experimenting with hypergraphs, an idea that we had already toyed with based on a visualization from a past project.

I thank Pedro Ribeiro for organizing Porto Winter School in Network Science, which was an event I thoroughly enjoyed, and for our very useful meeting, where he provided feedback on some of my ideas about node fatigue, as well as general hypergraph-based suggestions and references. I also thank everybody else that contributed towards this journey, that are not mentioned here.

I thank Bruno Martins, my external advisor, for his valuable input throughout this doctoral work and, in particular, for motivating the idea of TF-bins. And last, but not least, I thank my supervisor, Sérgio Nunes, for being a great mentor and a great friend, throughout the many years we have worked together. In the context of this thesis, I thank him for the suggestion to use a literature review and overall research management methodology based on a wiki, which was central to the organization of this doctoral work and the carried experiments. I have very much appreciated his critical thinking and pragmatism, which he brought to our weekly meetings, as well as his continuous availability, particularly during the hardest times. There are not enough words to thank Sérgio for being the calm in a sea of turbulence and for helping me reach a good port, both with my work and with my life. You have made me a better person. Thank you!

It really takes a village.

José Devezas was supported by research grant PD/BD/128160/2016, provided by the Portuguese national funding agency for science, research and technology, Fundação para a Ciência e a Tecnologia (FCT), within the scope of Operational Program Human Capital (POCH), supported by the European Social Fund and by national funds from MCTES.

Co-financed by:

REPÚBLICA PORTUGUESA | CIÊNCIA, TECNOLOGIA E ENSINO SUPERIOR

FCT Fundação para a Ciência e a Tecnologia

POCH

PORTUGAL 2020

UNIÃO EUROPEIA Fundo Social Europeu

To my father, who has always supported me in solving life's problems with calm.
And to his parents, my grandparents, who taught him humility and hard work.

To my mother, who is a great cook, and always motivated my intellectual development.
To my grandfather, my role model, who taught me the value of knowledge.
To my grandmother, a family woman, who was empathetic, considerate, and compassionate.

To my sister, who has always been a companion in fun and games.

To Sofia, who is my everything! ♡

And to Sérgio. Thank you for helping me move forward in my own way! :-)

*Your future hasn't been written yet. No one's has. Your future is whatever you make it. So make it a good one.*

—Doc., *Back To The Future*

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF LISTINGS

# CONTENTS

# GLOSSARY

| | |
|---|---|
| n-gram | A sequence of n consecutive terms. 40 |
| Army ANT | Army ANT[1] is a system developed during this doctoral work to compile all the developed code, with the explored index and search approaches, for entity-oriented search, and for multiple search tasks. It is based on Python, which integrates with external models written in other languages, mostly Java, but also R and C. 24, 102, 125 |
| arXiv | Open access repository of electronic pre-prints, which moderated but not peer reviewed. It covers work from the fields of physics, mathematics, computer science, quantitative biology, quantitative finance, statistics, electrical engineering and systems science, and economics. It counts with over 1.5 million deposited publications. 73, 76 |
| chunking | Chunking, or shallow parsing, consists of grouping parts-of-speech, in particular to identify noun phrases (NP chunking). 18 |
| cosine similarity | $sim(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\|\|\vec{b}\|} = \frac{\sum a_i b_i}{\sqrt{\sum a_i^2}\sqrt{\sum b_i^2}}$ 51 |
| dependency parsing | Syntactic dependency parsing, or deep parsing, identifies direct relations between words in a sentence, showing which words modify each other, or their syntactic functions (e.g., *SUBJ* for subject, or *ATTR* for attribute). 18 |
| Euclidean distance | $dist(\vec{a}, \vec{b}) = \sqrt{\sum (b_i - a_i)^2}$ 51 |
| evaluation forum | An evaluation forum is a scientific event organized with the goal of assessing the performance of models in a common experimental environment. In information retrieval, this usually means access to a common test collection, with a common set of topics illustrating an information need (transformable into a query), and relevance judgments, with manually assigned grades of relevance. 4 |
| heat kernel | $dist(\vec{a}, \vec{b}, \sigma) = \exp -\frac{\|\vec{a}-\vec{b}\|^2}{\sigma}$ 51 |
| hyperparameter | A hyperparameter is a configuration variable that can be set for a given learner algorithm. Conversely, the learner algorithm usually finds parameter values (e.g., the weights on a neural network), thus requiring these two levels of distinction. 39 |

1 https://github.com/feup-infolab/army-ant

inverted index
: An inverted index, or inverted file, consists of a mapping of terms to postings lists, where each posting contains information about a document that mentions the term. Stored statistics usually include the term's document frequency, the frequency of the term within each document and often as well the positions of the term within the document. The inverted index can be represented using different data structures. For instance, Apache Lucene uses skip lists for the inverted file and finite state transducers to map the indexed terms in memory. 18

Kendall tau
: Also know as the Kendall rank correlation coefficient, it corresponds to the fraction of the difference between the c concordant and d discordant pairs, over the maximum number of possible combinations: $(c - d)/(c + d)$. 39

MAiP
: The mean average interpolated precision is computed using the interpolated precision, at different levels of recall, instead of using simple precision. Refer to Gurajada et al. [1, Eq.5] for further detail. xiii

one-hot
: The one-hot representation is used in machine learning to numerically represent categories (or words) without assigning order. Instead of representing each category using an integer, a vector of size equal to the number of categories is used to represent a category by setting its corresponding position to one. 40

POS tagging
: Part-of-speech tagging consists of identifying each token with a label that assigns it a grammatical description (e.g., *NNP* for proper noun, or *RB* for adjective). 17

posting
: A posting identifies a document where a given term appears, storing statistics like the frequency of the term in the document, or the positions of the term in the document. The plural, postings, referes to the list of all postings for a given term. 105

qrel
: Relevance judgments file, in the TREC format, consisting of four fields: topic number, feedback iteration, document number, relevance score. 65, 84, 97

quad
: A quad is an extended triple, where the fourth element represents the URI (Uniform Resource Identifier) of the context or graph. 57, 105, 185

SIGARRA
: In Portuguese, SIGARRA stands for "Sistema de Informação para Gestão Agregada dos Recursos e dos Registos Académicos", which, in English, translates to "Information System for the Aggregated Management of Resources and Academic Records". SIGARRA is the information system used at the University of Porto. It is provided as a web site to each organic unit, which means that each faculty runs its own SIGARRA instance, as well as other institutional units, like the rectory, or SASUP, the University of Porto's Social Action Services. 24, 103, 107, 110

skip list
: Skip lists are lists that, in addition to a pointer to its next element, also provides shortcut pointers to jump ahead $n$ elements in the list. There are also multi-level skip lists, where shortcuts simultaneously provide jumps for different values of $n$. Inverted index technology, like Apache Lucene, takes advantage of this data structure to speed up the postings list intersection operation for the query terms. When compared to a regular linked list, which takes $O(n)$ time to check if an element exists, a skip list only requires $O(\log n)$ time. 105

skip-gram
: A sequence of terms separated by a given number of other terms. 38

Terrier
: Open source platform, written in Java, for carrying research and experimentation in text retrieval, providing easy integration of TREC and CLEF test collections. 7

test collection
: A test collection, in the context of this work, either refers to a dataset of text documents, or a dataset of combined data (e.g., text annotated with entity mentions, usually including document links and entity links from a knowledge base), with or without a fielded structure, accompanied by a set of topics representing an information need (directly or indirectly transformable into a query), and a set of relevance judgments, with manually assigned grades of relevance, that work as a ground truth. 4

TREC
: An evaluation forum that offers several thematic tracks for the development of information retrieval over text collections. Each track provides their own tasks for offline assessment over a common test collection, containing documents, topics and associated relevance judgments, although some tracks also rely on online assessment, most notably the OpenSearch track. xiii

triple
: A triple is a semantic statement containing a subject, a predicate, and an object, in this order. The subject and predicate are usually expressed as URIs, while the object can either be expressed as a URI or a literal. xv, 15, 46, 103, 149

triplestore — A triplestore is a database for storing linked data, in the form of triples or quads. Each triple contains a subject (S), a predicate (P) and an object (O). A quad extends a triple with a context or graph (G). A triplestore generally defines indexes like a SPOG or a POSG to improve the performance of SPARQL queries. 18

# ACRONYMS

| | |
|---|---|
| ACL | Annual Meeting of the Association for Computational Linguistics 76 |
| ACM | Association for Computing Machinery 74 |
| ANT | Ad hoc search of eNtities and Text 24, 102, 125 |
| AP | Average Precision 250 |
| API | Application Programming Interface 63, 106 |
| ARPA | Advanced Research Projects Agency 4 |
| bpref | Binary Preference 44 |
| BRAT | BRAT Rapid Annotation Tool 109, 111, 112 |
| CERN | Conseil Européen pour la Recherche Nucléaire 23, 53, 252 |
| CIKM | Conference on Information and Knowledge Management 73, 76 |
| CLEF | Conference and Labs of the Evaluation Forum 30, 60 |
| CoRR | Computing Research Repository 73, 76 |
| CRF | Conditional Random Field 109, 111 |
| DCG | Discounted Cumulative Gain 53 |
| DING | Dataset rankING 10 |
| DLN | Document Length Normalization 7 |
| DUL | DOLCE+DnS Ultralite 114 |
| ECIR | European Conference on Information Retrieval 76 |
| ELC | Entity List Completion 63 |
| ESA | Explicit Semantic Analysis 38, 50 |
| FEUP | Faculdade de Engenharia da Universidade do Porto 98 |
| FEUP InfoLab | Laboratory of Information Systems from the Faculty of Engineering of the University of Porto 125 |
| FTP | File Transfer Protocol 12 |
| GATE | General Architecture for Text Engineering 109 |
| GMAP | Geometric Mean Average Precision 84, 140, 222, 231 |
| GoE | Graph-of-Entity 155, 220 |
| GoW | Graph-of-Word 154, 220 |
| GPU | Graphics Processing Unit 252 |

| | |
|---|---|
| GSF | Groupwise Scoring Function 38 |
| HGoE | HyperGraph-of-Entity 181, 206, 220 |
| HITS | Hyperlink-Induced Topic Search 6, 8 |
| IDF | Inverse Document Frequency 4, 5, 7, 31, 232 |
| IE | Information Extraction 115 |
| INEX | INitiative for the Evaluation of XML Retrieval 10, 24, 30, 60, 62, 65–67, 81, 122 |
| IR | Information Retrieval 3, 4, 9, 30, 62–64, 70 |
| JASIST | Journal of the Association for Information Science and Technology 76 |
| JSON | Javascript Object Notation 96, 104 |
| JSON-LD | Javascript Object Notation for Linked Data 9 |
| KDD | SIGKDD Conference on Knowledge Discovery and Data Mining 76 |
| LODE | Linked Open Descriptions Of Events 114 |
| MAiP | Mean Average interpolated Precision 65, 66, *see also* MAiP |
| MAP | Mean Average Precision 39, 63, 65, 66, 84, 89, 126, 140, 222, 228, 231, 315, 318 |
| MART | Multiple Additive Regression Trees 38 |
| MMR | Maximal Marginal Relevance 221 |
| NDCG | Normalized Discounted Cumulative Gain 37, 39, 45, 50, 63, 231 |
| NDCG@p | Normalized Discounted Cumulative Gain at a cutoff of p 84, 140, 222 |
| NER | Named Entity Recognition 111, 222, 247, 334 |
| NERD | Named Entity Recognition and Disambiguation 15 |
| NIST | National Institute of Standards and Technology 4, 63, 97, 221 |
| NLTK | Natural Language ToolKit 109 |
| OWL | Web Ontology Language 105 |
| P@n | Precision at a cutoff of n 43, 84, 140, 222, 231, 318 |
| PCA | Principal Component Analysis 58 |
| PDLN | Pivoted Document Length Normalization 4, 6, 31, 232 |
| PhD | doctor of philosophy 11 |
| POS | Part-Of-Speech 7, 53, 109 |
| RDF | Resource Description Framework 8, 30, 39, 41, 46, 60, 149 |
| RDFa | Resource Description Framework in attributes 9 |

| | |
|---|---|
| REF | Related Entity Finding 63 |
| REMBRANDT | Reconhecimento de Entidades Mencionadas Baseado em Relações e ANálise Detalhada do Texto 109 |
| REST | REpresentational State Transfer 106 |
| RWS | Random Walk Score 131, 134, 182, 206, 219, 317 |
| SIGIR | Special Interest Group on Information Retrieval xx |
| SIGIR (conf.) | ACM SIGIR Conference on Research and Development in Information Retrieval 73, 76 |
| SJR | SCImago Journal Rank 71, 73 |
| SKOS | Simple Knowledge Organization System 46, 334 |
| SPARQL | SPARQL Protocol And RDF Query Language xvii, 15, 21, 30, 46, 48, 66, 104 |
| SQL | Structured Query Language 46, 104 |
| SSOAR | Social Science Open Access Repository 97, 98, 147, 156, 239, 241, 242 |
| SVM | Support-Vector Machine 39, 109 |
| TF | Term Frequency 4, 5, 7, 31, 232 |
| TF-IDF | Term Frequency $\times$ Inverted Document Frequency 5, 6, 31 |
| TREC | Text REtrieval Conference xv, 4, 6, 10, 24, 30, 60, 62–65, 67, 81, 122, 147, *see also* Text REtrieval Conference |
| URI | Uniform Resource Identifier xv, xvi, 67 |
| WaPo | TREC Washington Post Corpus 239, 241 |
| WSDM | ACM International Conference on Web Search and Data Mining 76 |
| WWW | International World Wide Web Conference 73, 76 |
| xinfAP | eXtended INFerred Average Precision 65, 66, 231, 250 |
| XML | eXtensible Markup Language 18, 19, 65, 66 |
| YAGO | Yet Another Great Ontology 48 |

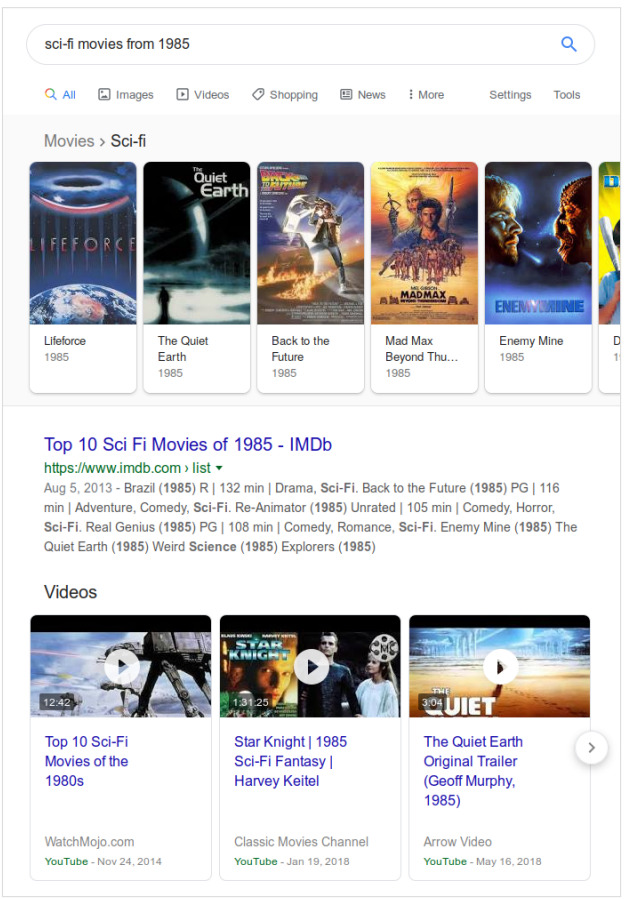
# 1 INTRODUCTION

## Contents

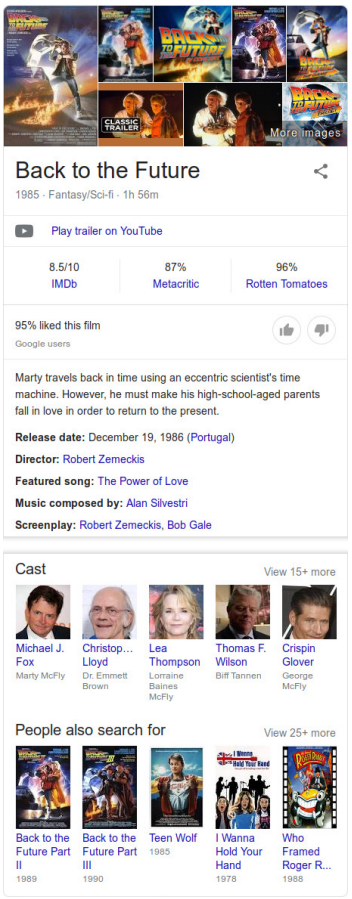


Search originated in the library, but it flourished in the web. It all started with the need to find the right content for the given information need. At first, there were directories, that were maintained to organize web sites into categories and to facilitate browsing. Then, as the web blossomed and site production increased, search engines became a necessary tool to reach the desired information among an ever-growing volume of content. Initially, web pages were treated as simple documents and search was mostly keyword-based. Ad hoc document retrieval was the main paradigm for finding web pages that best matched the given search terms. Later on, as the semantic web [2] emerged, the opportunity for search engines to also account for semantics in user queries and documents presented itself. Furthermore, the information needs of users were increasingly communicated as complex and verbose questions about specific objects [3]. As users were becoming more demanding, with nearly 39% queries directly referring to entities and 87% containing at least one entity [4], the need for entity-oriented search became evident.

Search engines like Google are already able to fairly understand queries, exploiting semantics to provide substantially improved and more direct answers to the users. As seen on Figure 1.1a, when searching for `[ sci-fi movies from 1985 ]`, it is now common to receive a list of movies instead of a simple ranking of documents based on whether they contain the query terms. Similarly, when searching for a particular entity, like `[ back to the future ]`, modern search engines frequently display a widget with information about the movie, sometimes including related queries. This is illustrated in Figure 1.1b, where we find the suggestions for `[ Back to the Future Part II ]` and `[ Teen Wolf ]`, a sequel to *Back to the Future* and a popular movie starring the same actor, respectively — in April 2017, the same query had displayed `[ Directed by Robert Zemeckis ]` and `[ Time travel movies ]`, illustrating the dynamics of




**(a)** Entity list for `[ sci-fi movies from 1985 ]`.




**(b)** Information about `[ back to the future ]`.

**Figure 1.1:** Web search, based on Google Knowledge Graph, ran on November 2019.

the system. We can also find queries that lead to other relevant entities, such as `[ Michael J. Fox ]` or `[ Christopher Lloyd ]`, which are a part of the cast, a relation that could only be found thanks to Google's Knowledge Graph [5].

Despite all the advancements that have been made in search, there are still several open challenges and unexplored opportunities. In particular, little work has been done on unified approaches to information retrieval. Is there a way to model heterogeneous data as a single useful representation for retrieval? Can we seamlessly combine and represent unstructured text from corpora and structured statements from knowledge bases, so that we can take advantage of all available information to provide the best answer to the user's information need? Is there a universal ranking function that can be used to generally solve retrieval tasks? Will a unified framework be more effective and efficient or does generalization represent a tradeoff with performance? These are many of the questions that we explore in this thesis, while developing graph-based entity-oriented search and proposing new approaches that take the potential of generalization into consideration.

The structure of this chapter is organized as follows:

- **Section 1.1** provides an historical perspective on the area of information retrieval, beginning at the library, and covering the evolution of search [§1.1.1]. It then analyzes the mixed history of the web, knowledge graphs and real-world networks [§1.1.2], and it describes the events that led to entity-oriented search [§1.1.3]. To help guide the reader, we provide three timelines that in-

clude relevant events for each area, focusing particularly on events that are pertinent to this thesis within that area.

- **Section 1.2** reinforces the need for a global perspective to coexist with a specialized approach in science, highlighting the importance of consolidating models. It shows examples of unified models in different domains of science, from physics to machine learning [§1.2.1], comparing a classical and a contemporary definition of IR (Information Retrieval) to motivate the need for a new, more general, approach to the area [§1.2.2]. It then demonstrates the success of graphs as a representation data structure in multiple domains [§1.2.3], and it closes by positioning this doctoral work in the intersection area of unified information retrieval, whose development is motivated through this contribution [§1.2.4].
- **Section 1.3** introduces the basic concepts of a unified model for entity-oriented search, discussing about combined data as a way to link unstructured and structured data [§1.3.1], and specifying the retrieval tasks for which a universal ranking function is proposed [§1.3.2].
- **Section 1.4** presents an overview of the problem description, specifying the tackled challenges and formalizing the problem as a thesis statement.
- **Section 1.5** tells the story of how the research work progressed, in order to provide context, clarify the motivation, and position this thesis [§1.5.1]. It then presents an overview of the document structure, describing the content of the remaining chapters of this thesis [§1.5.2] and its main contributions [§1.5.3].

## 1.1 HISTORICAL PERSPECTIVE

Ever since the 7th century BC, with the Royal Library of Ashurbanipal, that humanity has been collecting and organizing knowledge [6, §2.1.1]. It was there that we found what is perhaps the first categorization of materials, into six subjects: history, law, science, magic, dogma, and legends. Centuries later, around the 3rd century BC and inspired by Ashurbanipal, the Great Library of Alexandria was founded. For three centuries it is said to have accumulated around 700,000 scrolls, losing around 40,000 scrolls in the first-century BC to a fire during Caesar's Civil War. Initially stored as wedge-shaped cuneiform tablets and later on evolving to scrolls, documents slowly became what we know them to be today, leading to our present form of writing, to paper manuscripts and to the current era of digital encoding.

Throughout history, libraries have been used as a means to solve information needs, upholding knowledge and the development of mankind, by recording and preserving, for future generations to build upon the discoveries of previous ones. With an ever-increasing, harder to manage, amount of information, there has also been a constant evolution of storage and retrieval approaches. One of the early solutions for efficient retrieval was the peek-a-boo system of punch cards [7, §2.2], which established a Boolean logic over document subjects, thus being able to identify a set of relevant documents, that covered all the desired subjects, simply by overlapping cards and peeking through the holes using light. As technology evolved, a significantly larger amount of information started to be stored within computers and, even more, spread over the internet within one of the most important inventions of the 20th century, the world wide web.

Present information retrieval, as a computer science subject, was born in the 1950s, when Hans Peter Luhn started taking a statistical approach to characterize and describe documents within a collection. He used thesauri and indexes to encode documents in the machine, tackling the problem of search in a way that has now become classical [8]. The approach was, however, quite novel at the time. Years later, in the

1970s, another key moment would take place, with Karen Spärck Jones' contribution to the statistical interpretation of term specificity [9]. In complement to term exhaustivity, which had already been studied by Luhn, leading to the concept of TF (Term Frequency), the work by Karen Spärck Jones materialized into another central statistic of information retrieval, the IDF (Inverse Document Frequency). With TF, we would identify documents with a higher number of mentions to words relevant to our information need. With IDF, we could reduce the influence of terms that had little discriminative power — happening when too many documents mentioned a term for it to be useful for an automatic system to distinguish between documents. However, a final problem became clear during the third edition of TREC (Text REtrieval Conference) — an evaluation forum for Information Retrieval organized by ARPA and NIST. Given the varying length of documents in the test collection, it became clear that results were biased towards longer documents. This led Amit Singhal, Gerard Salton, Mandar Mitra, and Chris Buckley to propose the introduction of a new component, PDLN (Pivoted Document Length Normalization) [10, 11], which would fairly rank documents independently of their length. These three elements, TF, IDF, and PDLN, are now central concepts in most information retrieval models — search requires a match between a query and several documents, based on whether they share many words of interest (TF), that do not appear in too many documents (IDF), with longer documents regularized to account for a higher chance of terms repeating (PDLN).

### 1.1.1 Information retrieval and the evolution of search

In information retrieval, and particularly in search, the research focus has been in developing ranking models, to improve effectiveness, but also on proposing indexing approaches, to improve efficiency. Figure 1.2 illustrates the evolution of search over time, signaling the year when the most important representation and retrieval models appeared. Hiemstra [12] was used as a reference for retrieval models, and Zobel and Moffat [13] for representation models. Other, more recent, graph-based models were also considered [14–16].

We begin in 1847, with George Boole and his approach to logic [17], that later led to the Boolean retrieval model. This was the first model used in information retrieval, even prior to computers, for instance implemented in the peek-a-boo punch card approach that we described before. Then, in 1945, Vannevar Bush, postulated a device that he called the *memex* (from "memory" and "index"). In his article, «As We May Think» [18], he defined this device as follows:

> *A memex is a device in which an individual stores all his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility. It is an enlarged intimate supplement to his memory.*
>
> – Vannebar Bush, 1945

Bush proposed that "selection by association, rather than indexing" could be mechanized, providing a better match to the human mind, which also "operates by association". He stated that "the process of tying two items together is the important thing" and predicted that "new forms of encyclopedias" would appear "ready made with a mesh of associative trails running through them, ready to be dropped into the memex and there amplified". He also suggested that there would be "a new profession of trail blazers, those who find delight in the task of establishing useful trails through the enormous mass of the common record". Bush was a true visionary, but it would take decades of research to even approach his vision. In fact, today we are still working on strategies to augment text with entities, and we haven't yet figured out the best way to create a truly useful *memex*. We have hypertext and knowledge bases, but we only recently started working on bringing them together to support search, fully exploiting available resources to respond to our information

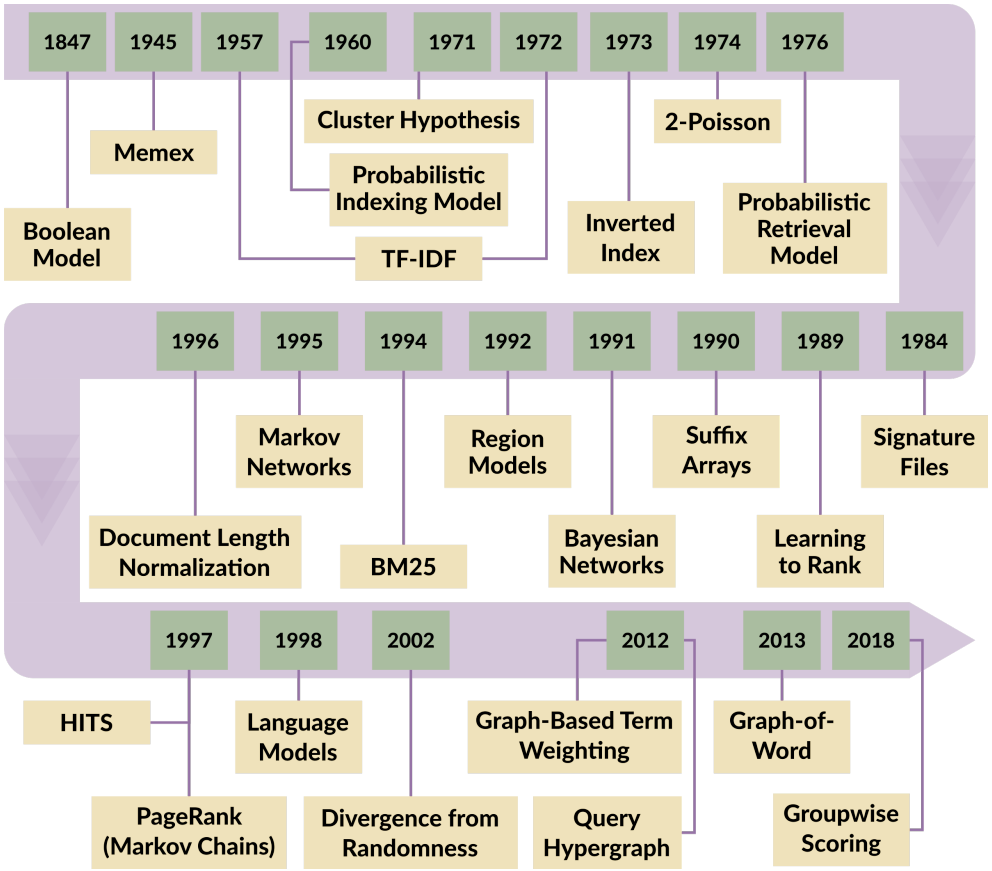



Figure 1.2: The evolution of search: a timeline of representation and retrieval models.

needs. The following paragraphs provide an overview on the history of information retrieval preceding this thesis, which is perhaps more aligned with Bush's vision of focusing on associations rather than present information retrieval trends.

In 1957, H. P. Luhn founded what we now know as information retrieval, with his statistical approach to search [8]. In 1960, his work was followed by the probabilistic indexing model [19], where the Bayes theorem was used as a weighting model to calculate the probability of a document given a term. This marked the beginning of two of the main approaches to ranking, the vector space model and the probabilistic models. In 1971, Jardine and van Rijsbergen [20] proposed the cluster hypothesis (see also Voorhees [21]), stating that "the associations between documents convey information about the relevance of documents to requests"). In 1972, Spärck Jones contributed to improving the discriminative potential of TF by proposing IDF [9], which was perhaps one of the most remarkable contributions to information retrieval, culminating in the classical TF-IDF ranking function.

At this time, given the infancy of the area, storage was not yet a central issue. Only in 1973, did the first indexing approaches for search emerge, with Knuth's clear definition of an inverted file [22, §6.5].

> ***Inverted files.*** *The first important class of techniques for secondary key retrieval is based on the idea of an inverted file. This does not mean that the file is turned upside down; it means that the roles of records and attributes are reversed. Instead of listing the attributes of a given record, we list the records having a given attribute.*
>
> – Donald E. Knuth, 1973

In 1974, the 2-Poisson model was proposed by Bookstein and Swanson [23] and further explored by Harter in 1975 [24]. It consisted of a linear combination of two Poisson distributions for the study of specialty words in technical literature. This

also introduced the concept of eliteness, where a subset of the collection showed evidence of a higher-than-normal presence of a given term, while a second subset did not. In 1976, Stephen Robertson and Karen Spärck Jones defied the principles put forward by H. P. Luhn. They proposed the probabilistic retrieval model [25], where the similarity between query and document was not the only considered criterion. Instead, they identified cases where the probability of retrieving a relevant document would be higher even if it did not contain a query term. This happened for example when the fraction of relevant documents was higher than the fraction of documents containing the desired term. Naturally, this required relevant documents to be known ahead of time, a case that is common, for instance, in the library, but not in the web.

In 1984, Faloutsos and Christodoulakis [26] were the first to attempt to compete with the inverted index, by proposing signature files as an alternative for indexing a collection. Their approach, however, required more disk accesses and space, and did not provide support for ranked retrieval. They also used probabilistic indexing, making false match elimination an issue. In 1989, learning to rank approaches emerged through Norbert Fuhr's work [27], providing an advancement over manually designing a function for combining and weighting features. This groundwork would, later on, lead to the automatic learning of the ranking function, over a wide range of document aspects (e.g., number of incoming links for a web page), as well as query-document relations (e.g., TF-IDF).

In 1990, there was a new attempt at an alternative to the inverted index, with Udi Manber and Gene Myers' proposal of the suffix arrays for string search [28], which, like signature files, failed to provide ranked retrieval, and were inefficient for large-scale applications. In 1991, Bayesian networks were used to define the inference network model [29], a graphical model describing dependencies between the information need, the query, the representation elements (e.g., terms) and the documents. In 1992, Forbes Burkowski took the initial steps towards region models [30], an extension of the Boolean model applied over segments of the text, showing that there was still room for non-ranking and set-based retrieval approaches. In 1994, at TREC-3, Robertson et al. [31] proposed Okapi BM25, a probabilistic model with two configurable parameters, $k_1$, for controlling the gain introduced by multiple occurrences of a term, and $b$, for controlling the influence of document length normalization. In 1995, Markov networks were proposed as an approach to information retrieval [32] by modeling the dependencies between queries and documents. Unlike Bayesian networks, there was no direction assigned to the dependencies within the graphical model for Markov networks. This meant that calculations were not done based on conditional probabilities, but instead on log-linear models of factors associated with the dependencies. In 1996, and thanks to the diverse lengths of documents in the TREC collection, Singhal et al. [10, 11] introduced Pivoted Document Length Normalization.

In 1997, link analysis emerged in information retrieval as way to take advantage of the hyperlinks connecting web pages. This was when the famous PageRank algorithm [33] appeared, proposed by Sergey Brin and Larry Page, the founders of Google. In that same year, Jon Kleinberg also proposed HITS (Hyperlink-Induced Topic Search) [34], as a way to exploit hyperlinks for search. While the prior could be computed offline for the web graph, the latter was applied over a subset of the web graph that contained previously retrieved pages relevant to the topic. In 1998, Ponte and Croft [35] proposed the inference of a language model for each document to estimate the probability of generating the query, taking language models from speech recognition and applying them to information retrieval. Ranking models like PageRank and language models both have an underlying graph, PageRank modeling transitions between web pages and language models modeling transitions between words. Identifying such regularities will pave the way towards a general model for information retrieval.

In 2002, Giambattista Amati, a member of the Terrier [36] team, proposed a new model, divergence from randomness, that built upon the 2-Poisson model and the concept of eliteness. It was based on the combination of a basic model (analogous to IDF), an after-effect, also called the first normalization (analogous to TF), and a term frequency normalization, also called the second normalization, or simply normalization (analogous to Document Length Normalization). Different models were provided as a choice for each of these three components.

In February 2012, Blanco and Lioma [15] structured several approaches for graph-based term weighting, over graphs that linked terms within a sliding window, optionally establishing direction based on POS (Part-Of-Speech) tags and Jespersen's rank theory [37] (from lowest to highest ranks). Based on the undirected and directed versions of these graphs, they proposed four ranking functions: *TextRank*, *TextLink*, *POSRank* and *POSLink*. *"Text"* or *"POS"* referred to the undirected or directed graph, respectively, while *"Rank"* or *"Link"* referred to the PageRank or indegree over that graph. In August 2012, Michael Bendersky and W. Bruce Croft took yet another step forward with their query hypergraph [14] for modeling higher-order term dependencies. Like Markov networks, a query hypergraph can also be transformed into, and solved through, a factor graph. However, unlike Markov networks, which only capture dyadic relations, the query hypergraph is able to capture polyadic relations. Finally, in 2013, building on the work by Blanco and Lioma, François Rousseau and Michalis Vazirgiannis proposed the graph-of-word representation model and the TW-IDF retrieval model [16], based on a directed graph to model term dependencies. They also used a sliding window, but this time it was not centered around each term, but rather started at the term, arguing that the following terms were more relevant than the previous ones for establishing a context. In 2018, in the context of learning to rank, and thinking about how loss can be pointwise, pairwise or listwise, Ai et al. [38] proposed that so should document scoring be computed groupwise. This defied the pointwise model that had been used so far to score documents (i.e., instead of computing $\texttt{score}(q, d)$, we should compute $\texttt{score}(q, D)$, considering a set of documents $D$ as opposed to a single document $d$ for a query $q$). This showed the importance of considering documents as a group, instead of individually, during ranking.

So far, we have learned about the evolution of representation and retrieval models mostly used in ad hoc document search. We have talked about how the web graph inspired PageRank, and how term dependencies were eventually represented as a graph and used for ranking documents. We have even identified a hypergraph-based model for capturing more complex dependencies. What we have not covered so far is entity-oriented search, but, before jumping into our main subject, let us first look at the evolution of graphs, in the web, as a way to represent knowledge, and as a tool to study real-world complex systems.

### 1.1.2 The web, knowledge graphs, and real-world networks

Graphs have had a significant role in the organization of documents and knowledge, but they have also been used to study complex systems. The web is a perfect example of the intersection and consolidation of the former concepts. When studying graph-based entity-oriented search, we must first understand how data is represented through graphs, as well as which statistics can be used for ranking within networks. Figure 1.3 illustrates the evolution of graphs, integrating important events from network science, as well as the world wide web, as it transformed into a more semantic web, mainly through knowledge graphs.

Graph theory was born with Leonhard Euler in 1735, when he presented his solution to the Seven Bridges of Königsberg. Centuries later, graphs became the mathematical structure used in network science to study real-world networks. Use cases ranged from the study of protein-protein interactions, coauthorship, or social networks, to the understanding of communication networks or the link ecosystem of

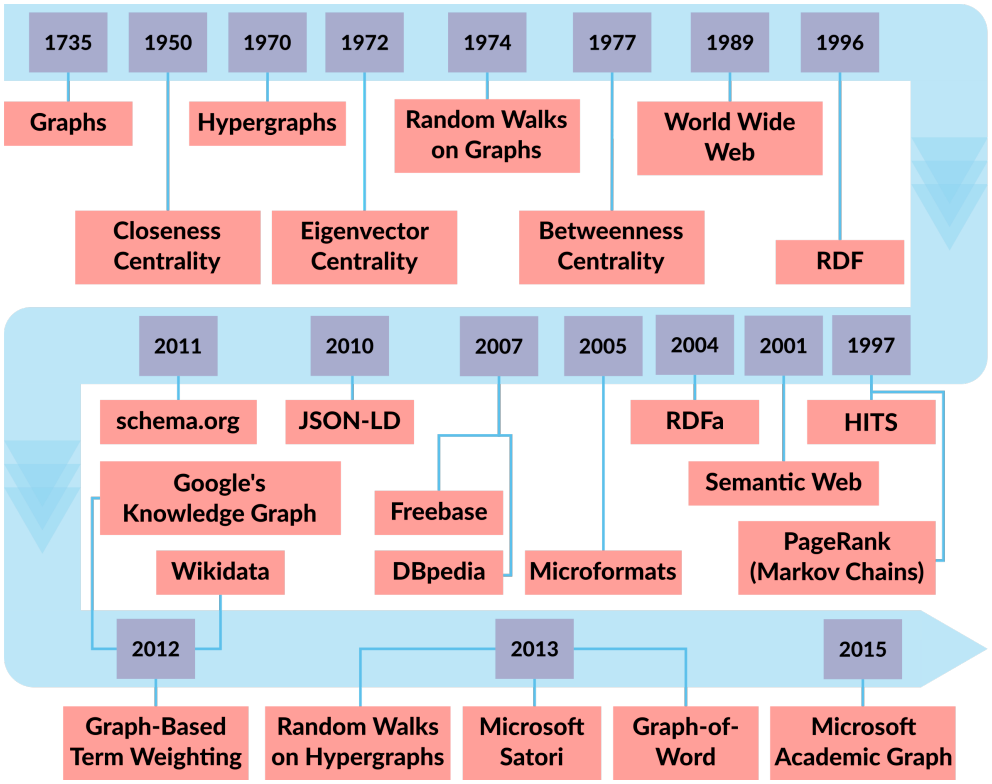



**Figure 1.3:** The evolution of graphs: a timeline of network science, the development of the semantic web and the rise of knowledge graphs.

the web. The first centrality metrics, built to measure node importance, originated in the 1950s, at a time when information retrieval was also in its infancy. It was precisely in 1950 that Alex Bavelas proposed the closeness centrality [39] as a way to assess communication scenarios within small groups as well as larger organizations.

And then hypergraphs emerged. Claude Berge, the founding father of hypergraph theory, wrote in the foreword of his 1973 edition of «Graphes et Hypergraphes» [40]:

> *At the Balatonfüred Conference (1969), P. Erdös and A. Hajnal asked us why we would use hypergraphs for problems that can be also formulated in terms of graphs. The answer is that by using hypergraphs, one deals with generalizations of familiar concepts. Thus, hypergraphs can be used to simplify as well as to generalize.*
>
> – Claude Berge, 1970

In science, we build models that try to explain reality, or parts of it. In this sense, a structure that is general and supports the easy representation of familiar concepts is worth considering over a structure with lower expressive power, even when there are equivalences. This thesis explores graph-based models, in which hypergraphs were included, with its central contribution lying in the latter.

In 1972, the eigenvector centrality was proposed by Phillip Bonacich [41, 42] to measure the power of an individual in the social structure and for predicting how fast information would spread. In 1974, a year after inverted files were clearly defined by Knuth, random walks on graphs emerged [43] and, in 1977, Linton C. Freeman's betweenness centrality provided a way to measure the importance of a node as a bridge [44]. In 1989, the web was born through Berners-Lee "Mesh" proposal [45] and, 7 years later, in 1996, RDF emerged [46]. The following year, PageRank [33] and HITS [34] were proposed, and 4 years later, in 2001, the semantic web was born [2].

In the years following the birth of the semantic web, semantic technology continued to evolve, with RDFa [47] appearing in 2004, to integrate entity semantics directly into web page attributes. In 2005, microformats [48] continued this trend and, in 2007, two important knowledge bases were born, Freebase [49] and DBpedia [50]. In 2010, JSON-LD [51] was proposed to represent linked data, and, in 2011, schema.org [52] emerged. In 2012, Google's Knowledge Graph [5] was born from Freebase, after Google acquired Metaweb. Freebase was shutdown in August 2016, leaving behind its resources to Wikidata [53], which also appeared in 2012, when graph-based term weighting was being proposed by Blanco and Lioma [15]. In 2013, Microsoft followed Google with their own knowledge graph, Satori [54]. 2013 was also the year when random walks on hypergraphs were formalized [55]. That same year, the graph-of-word was proposed by Rousseau and Vazirgiannis [16], and, in 2016, the Microsoft Academic Graph knowledge base was created [56]. Graphs have become ubiquitous, but spread differently across domains. As such, there is still much integration to be done, leading to better information retrieval, through the associations that Bush talked about in his idea for a *memex*.

We have looked at key events in the evolution of graphs, focusing on the representation of relations between entities, in particular for modeling knowledge, but also as a structure of relevance. As information retrieval's dependence on interrelating and cross-referencing information grew, so did its dependence on complex networks. With it, the access to the existing mechanisms of network science also extended the range of available tools to information retrieval. Could graph-based information retrieval be used for generally modeling and retrieving text and knowledge? Could it be the answer to a more general retrieval model, based on a universal ranking function? To reflect about this, let us now introduce and learn about entity-oriented search.

### 1.1.3 Documents meet entities: the birth of entity-oriented search

The shift to entity-oriented search was brought by the semantic web, information extraction, and information retrieval communities. The semantic web community wanted to find ways to exploit the machine understandable structure they had proposed to make knowledge more accessible. Thus, they experimented with the retrieval of subgraphs that would match a query graph [57, 58], or the retrieval of entities and relations to augment traditional search results [59, 60] — they modeled semantic search as either graph matching, or as ad hoc entity retrieval. On the other side of the spectrum, information retrieval scientists who had been, until then, focused on working with unstructured data from text, were now also considering structured data. With the help of information extraction approaches, they began introducing structure through semantic annotations in documents and queries, identifying entity mentions and linking them to the corresponding entities in a knowledge base. This led to the first approaches for ontology driven semantic search [61] — in IR, this was synonymous with ad hoc document retrieval, leveraging entities.

At this point, the concept of *semantic search* was becoming itself semantically ambiguous. In 2018, Krisztian Balog finally clarified the definition, using a more general approach that would subsume the previous definitions [62, §1.3.3]:

> *Semantic search encompasses a variety of methods and approaches aimed at aiding users in their information access and consumption activities, by understanding their context and intent.*
>
> – Kritsztian Balog, 2018

Figure 1.4 illustrates key events in the origin and evolution of entity-oriented search. It was in 2002, the year immediately following the birth of the semantic web, that the concept of semantic search started to be explored [57–60], with graph matching and ad hoc entity retrieval. It was also in 2002 that the «TAO of Topic

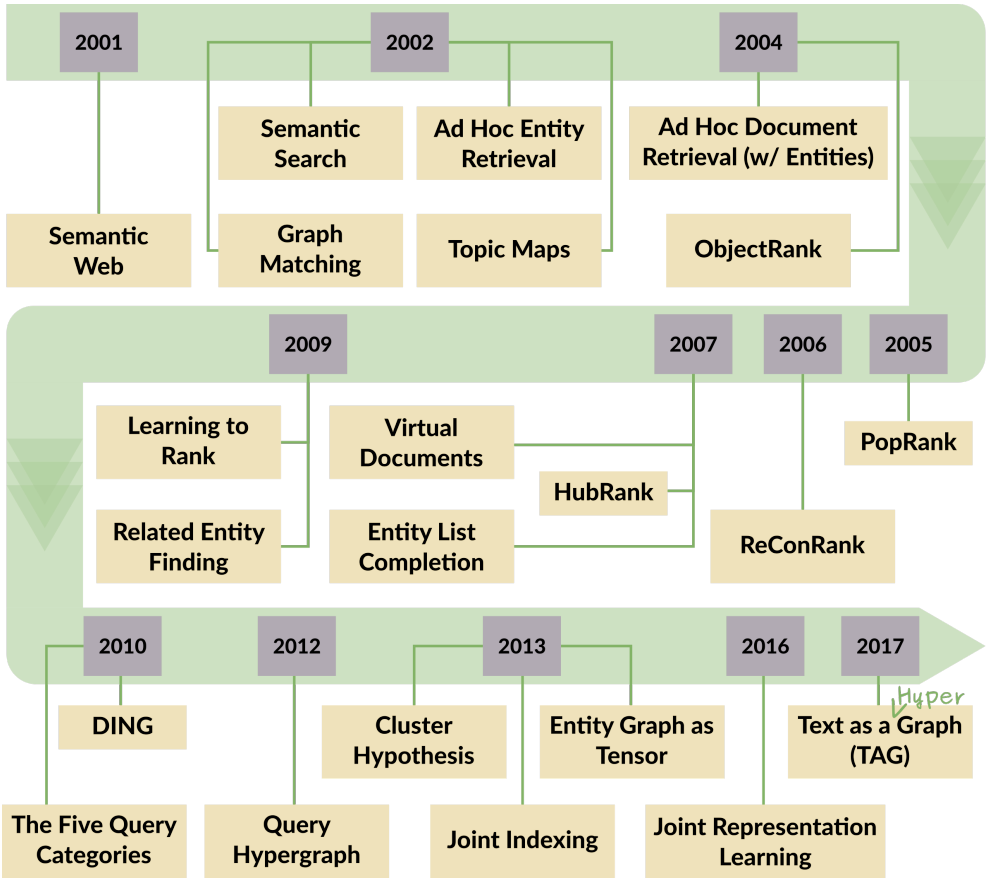



**Figure 1.4:** The evolution of entity-oriented search: a timeline of semantic retrieval approaches over corpora and knowledge bases.

Maps» was published [63], describing a structure that emerged in the 1990s based on HyTime [64], to describe topics, their associations and occurrences. This enabled multiple back-of-the-book indexes to be merged, establishing n-ary inter-topic links, leading to what was perhaps the first general representation model based on hypergraphs.

In 2004, the first methods for document retrieval that leveraged entities and their relations started to emerge [61]. In that same year, ObjectRank [65], the first PageRank application for ranking entities based on a keyword query was proposed. In 2005, Microsoft's PopRank continued this trend by combining web page links and entity relations to form a query-independent feature for Microsoft Academic Search (at the time still at its infancy with codename Libra). With a similar goal, in 2006, Hogan et al. proposed ReConRank [66], a combination of ResourceRank, computed over an entity graph based on subjects from triples, and ContextRank, computed over a context graph based on contexts linked by one or more common entities. Interestingly, the diagram they presented to illustrate their approach [66, Fig.2] resembled a hypergraph. Unfortunately, this data structure is frequently overlooked as a modeling solution — it is less known than graphs, but there is also a scarcity of tools for hypergraphs. In 2007, HubRank [67] was proposed as an approach for improved performance over ObjectRank. It was based on the precomputation of a set of hub nodes according to query logs and using random walk based fingerprints to estimate scores. Also in 2007, virtual documents were used for the first time [4] to represent and support the retrieval of entities. That same year, the task of entity list completion appeared in INEX 2007 [68], only coming to TREC in 2010 [69].

Two years later, in 2009, the task of related entity finding was introduced in TREC [70] and it was also around that year that learning to rank started to be applied to ad hoc entity retrieval [71]. In 2010, Delbru et al. [72] proposed DING

(Dataset rankING) as a way to rank datasets based on entity links, as well as inter-dataset links, and Pound et al. [3] proposed five query categories to classify ad hoc entity retrieval information needs: entity query, type query, attribute query, relation query, and other keyword query. In 2012, the query hypergraph model, which can loosely be considered a part of entity-oriented search, also contributed with a different view on modeling polyadic dependencies. Entities were defined as a part of the concepts, however the model did not contemplate knowledge base relations.

In 2013, the cluster hypothesis for entity-oriented search was proven by Raviv et al. [73], supporting the fact that entities similar to relevant entities also have a high chance of being relevant. In that same year, Bast and Buchhold [74] proposed a joint index for ontologies and text, essentially arguing that it would be the only viable way to cross-reference information transversal to unstructured and structured data during retrieval. It was also in 2013 that the community started experimenting with tensors to represent entity graphs [75], where a third dimension was used, for instance, for different predicates. Tensor factorization could then be used with listwise loss to obtain a learned ranking function. In 2016, representation learning was used to find a joint space for words and entities, reinforcing the importance of combining unstructured and structured data in a single "knowledge space". In 2017, Dekker and Birnbaum [76] proposed a representation model for text documents that would capture $n$-ary relations, with words forming sentences, sentences forming paragraphs and paragraphs forming documents, among others. Despite no direct relation with entities was established here, it is obvious that we might define entity markup annotations without significantly altering the model.

The refocus on associations, brought by graph-based models for information retrieval, has taken us full-circle to the original vision of Vannevar Bush and his *memex* device. With entities and their relations at the center of information retrieval, there is still, however, a need for intersecting unstructured and structured data, so that we can cross-reference information in both sources. In this thesis, we propose that this should be done using a general representation model and a universal ranking function over a graph or hypergraph data structure.

## 1.2 THE IMPORTANCE OF CONSOLIDATING MODELS

In the beginning, philosophy was an all-encompassing field of knowledge. Over time, however, as the body of knowledge increased in size and complexity, further specialization was required and new fields were born and developed. As fields like mathematics or information retrieval evolved, the view of the whole moved to the background, or rather became present only in sparse contributions across time — the ability to intersect knowledge, or the holistic aspects inherent to philosophy faded away with specialization. Information retrieval, for example, focused on "finding material [...] of an unstructured nature" [77], thus limiting its own ability to solve an information need. In the present, however, with search increasingly relying in heterogeneous collections, including corpora and knowledge bases, such a general view becomes once again relevant. Its importance to entity-oriented search lies in the need to cross-reference information from unstructured and structured data sources, to access previously unreachable information, benefiting from general models to better solve the information needs of the users.

In science, the process by which we develop general models has traditionally been a bottom-up process, from the modeling of individual phenomena, to the merging and generalization of the devised models — extending them to more than one phenomenon, or mapping the relations between phenomena. It is perhaps more than ever the function of PhDs (doctors of philosophy) in information retrieval to take a holistic stance towards the elements that lead users to the knowledge that they seek. The contribution of this thesis is on finding a general representation model for corpora and knowledge bases, as well as on defining a universal ranking func-

tion over such a model. In this section, we compare information retrieval with other domains of science, motivating the study of unified models, arguing for the usage of graphs as a supporting data structure for general models, and reinforcing the need for such an area of study in the information retrieval domain. This naturally positions this thesis in the area of entity-oriented search, where documents, entities and their relations coexist to solve information needs.

### 1.2.1 Unified models: from physics to machine learning

Proposing unified models requires a deep understanding of the phenomena under consideration, as well as of the already existing models for each phenomenon and their commonalities. Sometimes this means creating abstractions for the models, so that we establish a common ground at a higher level. Other times, it means specializing or further dividing the elements of the models, so that we find lower level connections.

Most fields of science have worked on the unification of theories or models in their area. In physics, Albert Einstein [78] has provided a unified description of gravity as a geometric property of space and time. In his last lecture, John von Neumann [79] established a parallel between the computer and the brain, motivating the cross-pollination between computer science and neuroscience. In cognitive science, Allen Newell [80] has proposed a unified theory of cognition, compiling a list of functional criteria that a human cognitive architecture should follow [81, Tab.1]. In information retrieval, Zhou and Huang [82] have explored the unification of text and visual features for image retrieval. In recommender systems, Bu et al. [83] have proposed a unified hypergraph that combined social and acoustic features for music recommendation. In information extraction, Moro et al. [84] have proposed a unified approach for entity linking and word sense disambiguation. In machine learning, Pedro Domingos [85] has worked on the unification of the master algorithms from "the five tribes of machine learning": symbolists, with inverse deduction; connectionists, with backpropagation; evolutionaries, with genetic programming; bayesians, with probabilistic inference; and analogizers, with kernel machines.

With the proliferation of entity-oriented search, the matter of a unified model to represent corpora and knowledge bases, as well as to retrieve any of the stored elements, becomes more evident. While some work has already been done regarding the joint representation learning, of words and entities [86], few attention has been given to this problem in entity-oriented search. Moreover, work regarding the unification of different retrieval tasks is quite limited — few of the available examples include, for instance, the unified modeling of information retrieval and recommender systems [87, 88].

### 1.2.2 Towards general approaches to information retrieval

In 1990, Alan Emtage [89] created Archie[1], the first internet search engine, built to locate content on public FTP servers. At that time, search was still heavily based on keyword queries, as inspired by the library and the search potential of the back-of-the-book index. However, with the evolution of the web and the devices used to interact with it, the materialization of people's information needs also evolved. Queries changed from simple topic-driven keywords to more complex entity-oriented structures. In 2007, Bautin and Skiena [4] found that nearly 87% of all queries contained entities, according to the analysis of 36 million queries released by AOL [90]. Furthermore, entities are also frequently found in documents — in the CoNLL 2003 English training set [91], there are 1.6 entities per sentence (23,499 entities for 14,987 sentences). Such a pervasive presence of entities, both in

1 http://archie.icm.edu.pl/archie-adv_eng.html

queries and in documents, easily justifies the current direction of search engines and their focus on entity-oriented search.

According to Balog [62, Def.1.5]:

> *Entity-oriented search is the search paradigm of organizing and accessing information centered around entities, and their attributes and relationships.*
>
> – Krisztian Balog, 2018

This presents a different side of the problem described in the classical definition of information retrieval portrayed by Manning et al. [77]:

> *Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).*
>
> – Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze, 2008

In entity-oriented search, the materials can be of an unstructured or structured nature. In fact, they are often a combination of both, either taking the form of semi-structured data or links between unstructured and structured data. In their survey on semantic search on text and knowledge bases, Bast et al. [92, Def.2.3] defined combined data as text annotated with entities from a knowledge base, or as a combination of knowledge bases with different naming schemes. Combined data is at the core of entity-oriented search. However, in the past, techniques for representing and querying corpora and knowledge bases have been explored separately. In a way, there are two different communities that require cross-pollination. Appropriately, Baeza-Yates et al. [93] had identified semantic search as a task that lies in between several areas of specialization. The same applies to entity-oriented search, which, according to Balog [62, §1.3.3], is subsumed by semantic search.

Modern search engines offer entity-oriented search through the orchestration of several components which are built on top of a common set of resources — a collection of documents and knowledge bases, containing terms and entities, along with links and resource statistics. A complete pipeline relies on components for entity ranking and similarity measurement, target entity type identification, word sense disambiguation and entity linking, document and query semantic analysis, query expansion and entity list completion, and query recommendation and related entity finding. Many of these approaches can be unified and refocused for better contributing to solving the information need. While there is a considerate challenge in reintegrating a set of components that have been strategically identified and developed over time, there might also be a considerate gain in breaking these boundaries, perhaps enabling uncertainty to be spread across the system rather than propagated from component to component. In this thesis, we motivate the idea of a more general approach to information retrieval through a unified modeling strategy. And, while not exclusively limited to, the task of merging many of these components can often be accomplished through graph-based models.

### 1.2.3 The success of graphs as general representation models

Graphs have had success in the representation of a wide range of data types. They have been used to represent documents, by establishing dependencies between their terms [15, 16], as well as knowledge bases, by modeling relations between their entities [49, 50, 53, 94, 95]. Recommender systems have also relied on graphs to generate music playlists [96], image retrieval has used them to establish similarity networks for reranking based on PageRank [97], and bibliometrics has relied on coauthorship network analysis to understand collaboration patterns and measure author importance [98]. Graphs have been used to pose representations of the brain [99], to study protein-protein interactions [100], to analyze social networks

and the semantic web [101], or even for counter-terrorism and intelligence [102]. Graphs have also supported the development of unified frameworks across different areas. Take for instance Moro et al. [84] who proposed a graph-based approach for unified word sense disambiguation and entity linking, Dietz [103] who proposed entity-neighbor-text relations to train a learning-to-rank-entities model based on edge feature functions, or even Richardson and Domingos [104], who proposed Markov logic networks, as a combination of probability and first-order logic, which could easily model the uncertainty of statements describing entity attributes and relationships. Graphs are even suitable to find explanations for entity relatedness, through the identification of relevant paths that describe how two entities are connected [105]. They are a flexible data structure in what comes to the representation of heterogeneous data and, in particular combined data. Yet, we can do even better, increasing the expressiveness of our representation model, if we use hypergraphs.

Hypergraphs are generalizations of graphs, that are polyadic instead of dyadic — i.e., hypergraphs support $n$-ary relations, while graphs only support binary relations. This means that they can be used to represent graphs, but also extended with relations that would otherwise require multiple edges, or auxiliary hub nodes, to be defined over a graph (e.g., synonymy, or contextual similarity). While in graphs two edges can be adjacent if they share a common vertex, in hypergraphs, two hyperedges can go beyond defining a simple adjacency relation, also modeling intersection or overlap. If we go back to the origins of information retrieval, to Luhn's illustration on the communication of ideas [8, Fig.1], we find the importance of modeling the overlap of common experience, at different levels of granularity. Luhn's model starts with an 'idea', the atomic element of communication, and progressively expresses it using: (i) a single sentence, for a high level of common experience; (ii) multiple sentences in a single paragraph, for a moderate level of common experience; and (iii) multiple paragraphs, with multiple sentences, for a low level of common experience. The expression of an idea requires overlap and a chaining of smaller ideas over a common experience. It is through the relations and overlap of experience that ideas are modeled. From this point of view, the hypergraph can be seen as a useful data structure for representing experience and ideas, usually expressed through text, entities and their relations. In Luhn's model, these relations are established by common experience, which can be seen as external knowledge that is required for a successful communication.

It would be a natural step to evolve a hypergraph-based retrieval model inspired by Luhn's model, for instance using documents to represent high levels of common experience, entities to represent the intermediate levels, and terms to represent the low levels. Perhaps the best reason to justify why this line of research was not pursed at the time is that the concept of hypergraph was only born 13 years later, in 1970, with the work by Claude Berge [40]. More recently, in 2017, adjacent representation ideas have been put forward by Dekker and Birnbaum [76], who explicitly explored hypergraphs to represent text, proposing several other relations like *:line*, *:phrase*, *:quatrain*, *:stanza*, or *:excerpt*. In information retrieval, hypergraphs as representation and retrieval models have been scarcely studied in the decades following Luhn's work, perhaps with the most relevant contribution being the already mentioned query hypergraph, by Bendersky and Croft [14, 106], who used hypergraphs as probabilistic graphical models to consider $n$-ary relations. Dietz [103] also touched on the subject of hypergraphs, but focused on the equivalent bipartite graph representation, instead of directly taking advantage of the hypergraph. When applied to information retrieval, hypergraphs have the potential to form a hybrid model that acts like a mixture of the Boolean model and graph-based models, simultaneously supporting set operations, like union and intersection, and graph-based operations, like the computation of shortest paths and node centralities. This is worth exploring, as it will takes us closer to a unified framework for information retrieval.

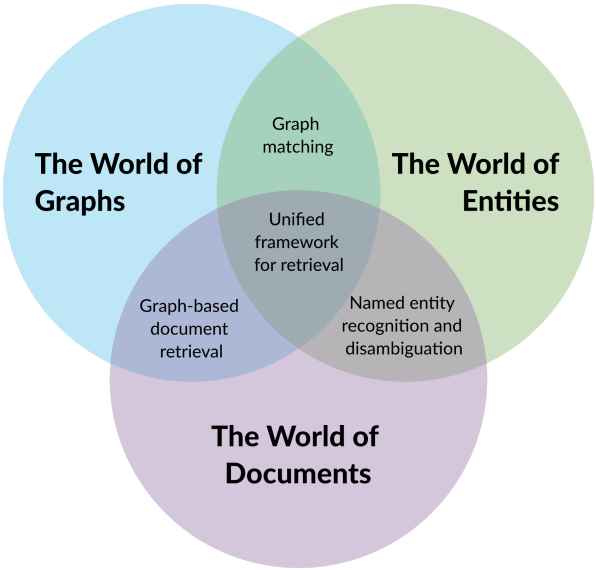



**Figure 1.5:** Graph-based entity-oriented search: a unified framework to index and search over combined data in the intersection of the worlds of documents, entities and graphs.

### 1.2.4 A unified framework for information retrieval

Figure 1.5 illustrates the rationale behind a unified framework for information retrieval, which we apply here to entity-oriented search. Corpora (the world of documents), and knowledge bases (the world of entities) intersect mainly in the task of semantic annotation. NERD (Named Entity Recognition and Disambiguation) is frequently used to identify segments of the text that mention a particular entity, usually taking advantage of the context, found both in the text and knowledge base, to link the mention to the correct entity instance. Some approaches to NERD even rely on graphs for disambiguation, as we have seen previously in Section 1.2.3 with Moro et al. [84]. Moreover, the world of documents and the world of entities each intersect with the world of graphs. Graph-based document retrieval is a good example of how graphs can be used to represent and retrieve documents, and graph matching is frequently used to query knowledge bases, usually through SPARQL (SPARQL Protocol And RDF Query Language), which establishes conjunctive graph patterns based on triples to be matched with the knowledge graph. With multiple overlapping points leading to graphs, the question that remains is whether we can devise a joint representation model based on graphs that supports the tasks required to solve an information need over text and entities (see the last paragraph in Section 1.2.2). Our goal is to prove that this is possible using graph-based entity-oriented search, opening the way for a new line of research in unified models for information retrieval, while also approximating adjacent areas like information extraction and linked data.

In entity-oriented search, data is heterogeneous. This means that the information leading to an answer that satisfies the information need might be spread across different sources. It can be explicitly described in knowledge bases, or implicitly available in corpora. Furthermore, it might require cross-referencing across data sources for an answer to be found. No matter the type of item to be ranked, we can benefit from linking the data to maximize the available information at any given stage of the retrieval process. In this particular case, we have two main units of retrieval — documents and entities — which can both be linked among themselves in homogeneous networks (i.e., modeling document–document and entity–entity relations). Furthermore, documents (or parts of documents) can also be linked to the entities that they mention, forming an heterogeneous network. Finally, entities are naturally linked to related entities within the same knowledge base, and they

can also be linked to other instances, representing the same entity within another knowledge base. These connections are the essence of combined data — they establish links between different sources of information, be it structured or unstructured. In this thesis, we propose a unified framework for entity-oriented search, first based on graphs, and then evolving to hypergraphs, both acting as general representation models for corpora and knowledge bases. We then design a universal ranking function, over our hypergraph-based model, that supports four different retrieval tasks, as a way to prove our thesis and motivate further research in unified models for information retrieval.

## 1.3 A UNIFIED MODEL FOR ENTITY-ORIENTED SEARCH

As we have seen, entity-oriented search not only encompasses tasks based on entity ranking, such as ad hoc entity retrieval, related entity finding, and entity list completion, but it also covers ad hoc document retrieval, as long as it relies on entities for semantic enrichment [62, Ch.8]. While these tasks can be modeled individually, they share a common collection of combined data, bringing together text and entities, in their heterogeneity, through annotations that connect mentions to entities, as well as individuals representing the same entity. A data structure capable of representing such heterogeneous data is a graph, which is why this thesis focuses on exploring graph-based entity-oriented search. Graphs have the ability to represent documents, entities, and their relations, working as a joint representation model that provides the opportunity to tackle information retrieval in a more general way.

In this section, we introduce combined data as the main type of dataset for entity-oriented search, showing how different inputs and outputs can lead to different retrieval tasks. We also present an overview on the main tasks in entity-oriented search, covering ad hoc document retrieval, ad hoc entity retrieval, related entity finding, and entity list completion.

### 1.3.1 Combined data: linking corpora and knowledge bases

The concept of combined data has been inherently present in information extraction, through entity linking, where textual mentions are identified, disambiguated and associated with the correct, or most probable, entity in a knowledge base. It has also been present in the semantic web, through ontology alignment, where the correspondence between instances in different ontologies is established by a relation of equivalence (e.g., *owl:sameAs*). The definition proposed by Bast et al. in their survey on semantic search [92, Def.2.3] accounted for these two aspects, distinguishing between *link* and *mult*, to refer to text-to-entity, and to cross knowledge base entity-to-entity relations, respectively. A slightly higher-level reiteration of that definition, also considering links between documents, is proposed:

**Definition 1.** *Combined data is a collection of corpora and knowledge bases, which includes not only the natural relations between documents (e.g., hyperlinks in the web), and entities (e.g., object properties in triplestores), but also cross-context relations, from mentions found in documents to entities in knowledge bases, and from entities found in knowledge bases to instances of the same entity in other knowledge bases.*

Figure 1.6 illustrates the usefulness of combined data in providing a common ground for integrating text and entities. As we can see, in yellow, there is a collection of documents (*C1*), with hyperlinks connecting them. There are also two knowledge bases (*KB1* and *KB2*), containing entities and their relations in green. In pink, we find the cross-context links that are crucial to combined data, connecting entity mentions, in documents, to entities in a knowledge base, or two instances of the same entity that appear in different knowledge bases. These cross-context

or annotation links act as access points to augment not only text with knowledge, but also knowledge with text. Take for instance *Doc.2* and *Doc.4*, which are not directly linked. We know, however, that there is a relation between these two documents, since they are co-cited by *Doc.1*. Additionally, based on the links from *Doc.2* to *Entity 1*, and from *Doc.4* to *Entity 1* and *Entity 2*, we now know that the two documents are related because they cite a common entity. This is reinforced by the fact that *Doc.4* cites *Entity 2*, which is related to *Doc.2* through *Entity 1*. As another example, let us now look at *Doc.3*, which mentions *Entity 5*. Since we also know that *Entity 5* and *Entity 7* refer to the same entity, then we might also reach *Entity 8* as a second degree connection. These links also contribute to measuring node importance. If we look at *Entity 1* without considering annotation links, it only has one incoming link. Thus, we might consider it less important than *Entity 4*, which has two incoming links. However, when also accounting for information from documents, *Entity 1* might be considered more important than *Entity 4*, since it has three incoming links, one from *Entity 2* and two other from *Doc.2* and *Doc.4*.

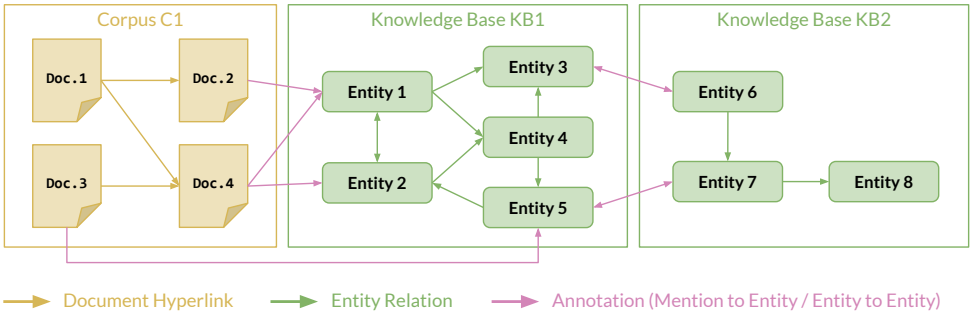



**Figure 1.6:** Integrating unstructured and structured data through corpus ↔ knowledge base and knowledge base ↔ knowledge base links.

Bast and Buchhold [74] presented the following example that illustrates the need for cross-referencing information from unstructured and structured data:

> *Consider the query for entertainers that are friends with [an astronaut who walked on the moon . . . ] and that the fact about friendship is retrieved from the text and not part of our ontology. There is no way to process the full-text and ontology part independently and afterwards combine the results.*
>
> – Hannah Bast and Björn Buchhold, 2013

In this particular instance, the ontology clearly identifies astronauts and entertainers. However, there can be multiple reasons why the friendship relation is not present in the knowledge base. This might be because the curator did not add it, or because the relation extraction system was unable to identify a friendship relation in the text, or even because that particular ontology does not model friendship relations. There are multiple solutions to this problem. We can extend the ontology, eventually reusing a different ontology to express friendship, and we can ask the curator to start considering friendship relations. We can also rely on open information extraction [107], which, without the need to define a vocabulary for specific relations beforehand, should be able to identify friendship relations automatically, when available in the text. However, these alternatives all have a cost. It represents an increase in the workload of the curator, a revision to the relation extraction system so that it supports additional relations without compromising accuracy, and a revision to the ontology. Even with open information extraction, which is perhaps the best option, there are also some concerns. Namely, it usually requires syntactic and lexical constraints to ensure a higher level of precision and recall, and a better tradeoff between those two metrics [108]. Achieving this can be computationally expensive, specially for large collections, since it usually requires POS tagging,

chunking, and dependency parsing [108, §5.1]. Moreover, extracting information from the combined data collection to be indexed is a process that represents a substantial modification to the original data, along with a potential loss of information introduced by additional uncertainty. The argument is not that this uncertainty should be removed, since, regardless of the approach, it is bound to be a part of the process. The argument is that, for information retrieval, uncertainty should be introduced only after considering the user's information need. This way, we ensure minimal loss of information — we take into consideration the user's query, all available text, and all available knowledge, and we then take advantage of a large number of relations to disambiguate and rank. The motivation is that small leads, if modeled correctly, can be as good for the retrieval process as a properly curated knowledge base for a specific domain.

As we have pointed out before, there is a clear advantage of using combined data to better solve information needs. By cross-referencing information from corpora and knowledge bases, we are able to explore new paths that provide further insights through previously unconsidered connections. This expands search possibilities and, at the same time, it also provides an approach for measuring importance through network structure. While conceptually sound, there is a challenge with representing and exploiting the relations within combined data, for search. The indexing of combined data has largely relied on the inverted index, and the triplestore, either losing the statistics of the inverted index used for ranking, or the complex relations stored within the triplestore. Another approach has been to separately compute and merge signals and lists from either storage system. Regardless, there is always a compromise that will prevent the usage of all available information to solve the information need. In this thesis, we propose a joint representation model for terms, entities and their relations that attempts to take a step towards solving this problem.

### 1.3.2 Retrieval tasks: towards a universal ranking function

In order to better understand the representation requirements for combined data and the suitability of graphs for supporting retrieval, we must understand the tasks in entity-oriented search. In this section, we present two different approaches for solving the user's information need, comparing classical text-based search and entity-oriented search. We also describe the four tasks that we consider fundamental in entity-oriented search, illustrating with examples.

Figure 1.7 portrays two approaches to solving an information need, based on different subclasses of combined data (structured, unstructured and semi-structured). As we can see, the basic approach to information retrieval is to build a retrieval model based on text corpora, web pages or XML. Then, for a given information need, usually expressed as a keyword query, we provide a list of ranked documents to the users. If the users were looking for a specific document, like a web page they want to access, then the system did its job. However, if the users were looking for a specific answer, they might have to read through the retrieved documents until they find the answer and solve their information need. Entity-oriented search takes advantage of knowledge bases, frequently in the form of an entity graph, to more effectively provide answers to the users. It may sill return documents, but it always takes advantage of the semantics imposed by the entities mentioned in the documents and their relations, to better measure relevance and more accurately provide an answer. It might also directly provide entities to the users, displaying an information card with metadata about a single entity (e.g., `[ back to the future ]`), or displaying a list of entities, if the query so requires (e.g., `[ sci-fi movies from 1985 ]`). Related entities might also be suggested in connection to one of the retrieved documents or entities.

In order to tackle each of these problems, we have segmented entity-oriented search into four fundamental tasks: ad hoc document retrieval, ad hoc entity re-

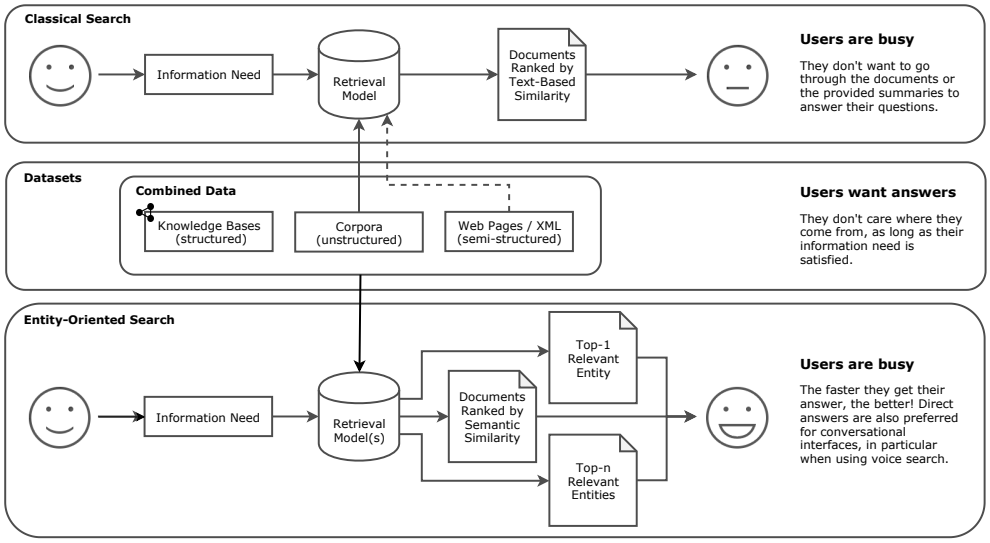



**Figure 1.7:** Solving an information need in classical search versus entity-oriented search (the dashed arrow symbolizes a partial dependency — e.g., discarding some of the structural information, such as the hierarchical relations in XML).

trieval, related entity finding, and entity list completion. A description of each of these tasks is briefly presented next.

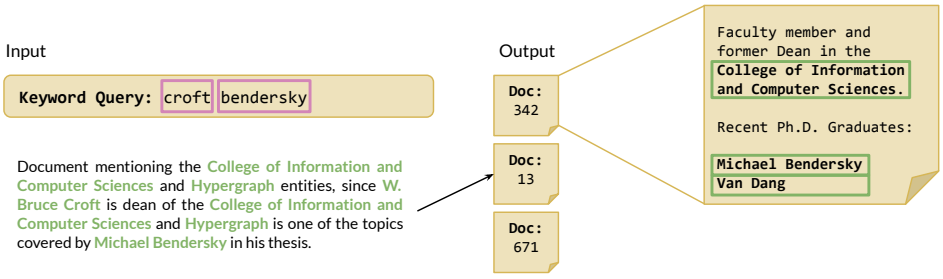



**Figure 1.8:** Ad hoc document retrieval, leveraging entities.

**AD HOC DOCUMENT RETRIEVAL** Figure 1.8 illustrates the traditional task of ad hoc document retrieval, where, given a keyword query, the search engine must retrieve a ranked list of documents, according to their relevance to the query. In text-based search, this is usually done by tokenizing each document and creating an inverted index where terms point to the documents containing it, along with several statistics. In entity-oriented search, however, there is also the question of how to integrate the information from documents and entities. Should we index them together as text? Should we compute signals from the inverted index and knowledge base and combine them? Is there a way to jointly represent documents and entities? How can we ensure we harness the complex relations of entities to improve retrieval effectiveness? The example in Figure 1.8 shows how entities and their relations can be used to influence and improve the ranking of documents. Given the query `[ croft bendersky ]`, we assume there is a query entity linking process that associates *"croft"* with *W. Bruce Croft* and *"bendersky"* with *Michael Bendersky*. Analogously, we assume that there is an entity linking process that associates each mention of an entity in the documents with its corresponding entity in a knowledge base. One way entities could then affect the ranking process would be by boosting documents with entities related to the query, not necessarily mentioned explicitly. In the example, a document mentioning *College of Information and Computer Sciences* and *Hypergraph* was boosted because these entities were related to *W. Bruce Croft* and *Michael Bendersky*. Se-

mantics was considered based on the entity graph and the *corpus* ↔ *knowledge base* relations.

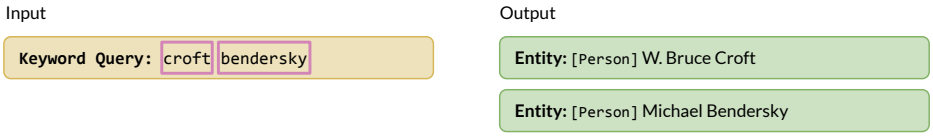



**Figure 1.9:** Ad hoc entity retrieval.

**AD HOC ENTITY RETRIEVAL** Figure 1.9 illustrates the task of ad hoc entity retrieval, where, given a keyword query, the search engine must retrieve a ranked list of entities, according to their relevance to the query. The challenge lies in the mismatch between the graph-based representation of the knowledge base and the multi-field document representation of the inverted index. Should we index the text associated with the entities? And, if so, which attributes and related entities should we consider, if any? Immediate neighbors, or more than that? Should we simply compute relevance weights directly based on the knowledge graph? In the example, *"croft"* and *"bendersky"* are the query terms used to search over the knowledge base, retrieving two tied entities, *W. Bruce Croft* and *Michael Bendersky*. While in ad hoc document retrieval we are required to leverage entities for it to be considered an entity-oriented search task, in ad hoc entity retrieval, the reverse is not true. While we are not required to leverage documents to improve entity ranking, this analogous step ensures that all available information is exploited, having the ability to improve performance [109].

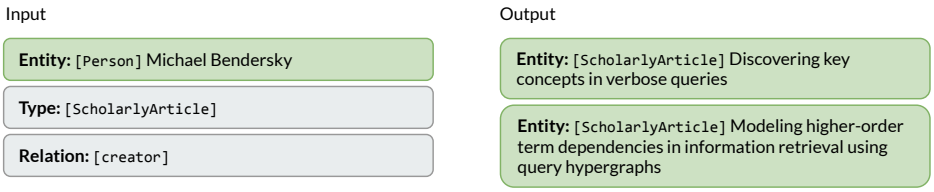



**Figure 1.10:** Related entity finding.

**RELATED ENTITY FINDING** Figure 1.10 illustrates the task of related entity finding, which is somewhat analogous to entity recommendation, without a user profile (a pseudo user profile is instead established by the query). Given an entity, a target entity type and a relation, the goal is to retrieve and rank other entities that respect the relation and target type. Traditional recommendation techniques, which also include graph-based approaches, can be used to support this task (e.g., Reinanda et al. [110] have experimented with both learning to rank and subgraph propagation in a Bayesian network). In the example, given the *Michael Bendersky* entity, we want to find scholarly articles created by him. Retrieved results include an article about *"verbose queries"* and another one about the *"query hypergraph"*.

**ENTITY LIST COMPLETION** Figure 1.11 illustrates the task of entity list completion. Similar to related entity finding, it also requires an entity, target entity type and relation as the query. Additionally, it considers example entities, as relevance feedback to inform retrieval. In the example, the same two articles shown in Figure 1.10 were retrieved, however the article about the *"query hypergraph"* was ranked higher, since the example article provided was also about the same topic.

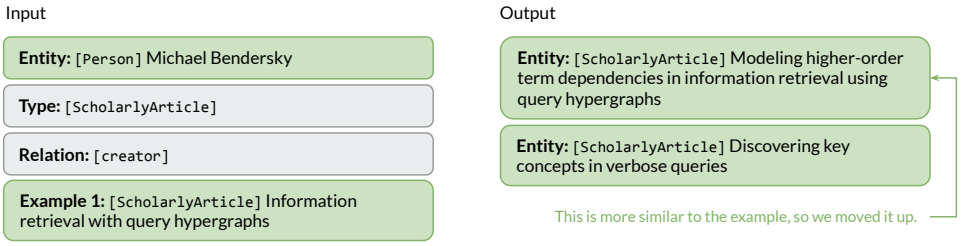



**Figure 1.11:** Entity list completion.

The four retrieval tasks can be described over a common collection of combined data and, were it not for the heterogeneity of unstructured corpora and structured knowledge bases, they could all be modeled simply through different combinations of input and output. For a keyword query and a ranking of documents, we would get ad hoc document retrieval. For a keyword query and a ranking of entities, we would get ad hoc entity retrieval. For an entity query with a single entity and a ranking of entities, we would get related entity finding. And, finally, for an entity query with multiple entities and a ranking of entities, we would get entity list completion. This is central to the motivation of this thesis, where we propose a representation and retrieval model capable of indexing combined data, as well as a universal ranking function that, solely by controlling input and output, is able to respond to any of these four tasks. Our generalization is applied only to existing tasks, however we could just as easily define a task where an entity query is used as input and a ranking of documents is used as output, obtaining the most relevant document for a given set of entities. This is an advantage of unified models.

## 1.4 PROBLEM STATEMENT

When answering a user's information need, entity-oriented search reconciles results from unstructured and structured data. This problem is frequently approached by establishing separate tasks, where the information need is solved as a combination of different subsystems. While each subsystem can use information from the other subsystems, they usually have their own central representation and retrieval model. For example, the inverted index is one of the main representation models in ad hoc document retrieval. And while structured information can be integrated into the inverted index to improve retrieval effectiveness, the rich and complex relations from knowledge bases are seldom transposed to the inverted index in an effective manner. For instance, related entities can be represented as text through a description or a profile. This way they can then be indexed in one or multiple fields of the inverted index and contribute to the ranking function as any other field would. Another approach is to separately query the inverted index and the knowledge base and combine document and entity weights. This means that the approach has been to either combine the output of two models, or to translate one type of data to approximately fit the model of the other type of data, from then on working exclusively in that domain. Both approaches represent a missed opportunity to cross-reference units of information from unstructured and structured sources (see also the example cited in Section 1.3.1). A similar case can be made for knowledge bases where indexes over triples can be queried through SPARQL, sometimes taking advantage of full-text search to filter fields. There is clearly an opportunity for a joint representation model, with ranking approaches that are generalizable to different units of information, and to different tasks over those units. With this thesis, we focus on developing the groundwork for such a model.

Although there is already work where unstructured and structured data are combined, models have been overly centered on one or the other type of data, frequently considering one of them as the external signal. It is in this lack of a balanced middle ground that we find the opportunity for a contribution. The hypothesis is that, by proposing a representation and retrieval model where text and knowledge are seamlessly considered, we will be able to:

1. Jointly represent terms, entities and their relations in a single index;
2. Propose a universal ranking function for multiple entity-oriented search tasks;
3. Improve overall retrieval effectiveness through the unification of information sources.

More formally, this threefold hypothesis is expressed by the following statement:

**Thesis statement**

A graph-based joint representation of unstructured and structured data has the potential to unlock novel ranking strategies, that are, in turn, able to support the generalization of entity-oriented search tasks and to improve overall retrieval effectiveness by incorporating explicit and implicit information derived from the relations between text found in corpora and entities found in knowledge bases.

## 1.5 THESIS OUTLINE

In the journey to prove the thesis statement described in the previous section, different approaches were explored. Before going further into this work, we believe it is useful to provide a brief overview of the rationale behind the line of research that we followed. We then identify the main reasons to read this thesis, as well as the main contributions of this work, guiding the reader to the respective chapters and sections.

### 1.5.1 The story

We began by studying the literature and developing a prototype for an entity-oriented search engine (Section 5.1). This provided insight into the challenges and opportunities in the area. Once again, we studied the literature (Chapter 2), which led to the graph-of-word [16], a graph-based model for ad hoc document retrieval, that could also be used for keyword extraction [111]. We then created an experimentation and evaluation platform, that acted as a central codebase for the contributions developed during the doctoral work (Section 5.2). Every experiment and script that we prepared is a part of this platform, called Army ANT, which is publicly available as an open source project in GitHub[1]. Our first approach, the graph-of-entity (Chapter 6), was inspired by the graph-of-word. The main focus, however, was on integrating entities, which we did by linking to terms that matched a part of the entity's name. We kept term relations, but only for immediately adjacent terms (i.e., we used a window size of 1), and we also kept entity relations based on the knowledge base. We transformed the document-based graph into a collection-based graph, identifying different documents by adding a *'doc_id'* attribute to each entity node — this was only possible because we were working with the INEX 2009 Wikipedia collection [112] (Chapter 4), where a document is always associated with an

1 https://github.com/feup-infolab/army-ant

entity. On one side, because the graph-of-entity would not explicitly represent documents, and, on the other side, because the number of edges grew exponentially with the number of nodes, we started thinking about alternatives. Additionally, we wanted to experiment with synonyms and contextual similarity, which meant that the number of edges would once again increase substantially, making it a priority to improve performance, even if only to make experiments viable. Since it was becoming so hard to deal with the growth in the number of edges, we thought we could take advantage of hypergraphs to create relations based on sets of nodes instead of relying only on dyadic relations (e.g., synonymy could be represented through a single hyperedge). Thus, we reworked the graph-of-entity, proposing the hypergraph-of-entity, which has become the central contribution of this thesis (Chapters 7, 8 and 9).

At that point, we were able to represent a higher number of relations. However, for the hypergraph-based model, it was not viable to compute the entity weight — the ranking function used in the graph-of-entity, based on shortest distances. This led us to experiment with approximated techniques to capture network structure. We ended up exploiting random walks for the task of node and hyperedge ranking over the hypergraph. In turn, this led us back to link analysis, to PageRank [113], at the origins of graph-based algorithms for ranking. Thus, a detour to study PageRank variations was justified (Appendix A and Section 2.2.8), learning about power iteration and how to reduce each variation to a power iteration solution. At that time, inspired by von Neumann's last lecture [79] and the idea of neuronal fatigue, we proposed random walks with fatigue, first over the hypergraph and then as a Fatigued PageRank for graphs (Appendix B). As a long-term approach, we thought that we would be able to exploit Multilinear PageRank [114] to adapt Fatigued PageRank for hypergraphs — this was dependent on finding a tensor-based representation for our hypergraph. However, we found that there is a considerable challenge in representing a general mixed hypergraph using tensors. The main work we identified on this topic was developed at CERN and at the University of Geneva, by Ouvrard et al. [115], but the challenge they addressed was on representing general hypergraphs, considering only undirected hyperedges. Their work was based on homogeneous polynomials to add to lower cardinality hyperedges, until a tensor for a uniform hypergraph could be coded.

Contributing to that work was outside of the scope of this thesis, making this a nonviable research line. We opted, instead, to focus directly on simulating random walks over the hypergraph, step by step, in order to measure the effectiveness of our approach, regardless of efficiency. This was the approach that supported several iterations over the random walk score, the universal ranking function that we propose. The random walk score is largely representation-driven, in the sense that the structure of the hypergraph is directly responsible for the quality of the ranking. Nevertheless, there were multiple parameters that we could study, and we could also experiment with different index extensions, reconfiguring the representation model and studying the effects in ranking performance, without altering the ranking function. From that moment on, we focused on the hypergraph-of-entity, studying it (Chapter 8) and evaluating several models based on different configurations of the hypergraph-of-entity (Chapter 7), as well as the random walk score, which led to a model for general entity-oriented search (Chapter 9).

### 1.5.2 Document structure

In order to better convey the work that we just described, we organized this document according to the following structure:

- **Part i** covers the state of the art on graph-based entity-oriented search, mostly through subjects that potentiate this approach, due to the limited number of contributions to this specific topic.

  - **Chapter 2** surveys the state of the art on graph-based entity-oriented search. It covers classical text retrieval and learning to rank, and it explores graph-based models for retrieval, both leading to specific applications for entity-oriented search. It then examines evaluation forums and test collections, and it closes with a discussion, presenting observations on the subject, as well as an organized overview on the covered approaches.

- **Part ii** introduces the materials and methods used throughout the thesis.
  - **Chapter 3** covers the approach to literature review, introduces the empirical cycle and the particular instantiation used in this work, and describes the approach to systematic documentation based on a wiki.
  - **Chapter 4** describes the datasets used throughout this thesis, including test collections for retrieval, produced and published datasets, and auxiliary datasets used for secondary tasks.
  - **Chapter 5** describes the software developed throughout this doctoral work, focusing on ANT and Army ANT. ANT is an entity-oriented search prototype developed to index and search over the public content provided by SIGARRA, the University of Porto's information system. Army ANT, is a workbench for innovation in entity-oriented search, providing a framework for implementing and testing search engines, easily reusing test collections, configuring ranking functions and their parameters, and learning about collections, retrieval models, and the ranking functions.

- **Part iii** describes the main contributions of this doctoral work, prepared with the goal of proving the thesis.
  - **Chapter 6** describes the graph-of-entity model, as well as the experiments carried over the INEX 2009 Wikipedia collection and the TREC 2017 OpenSearch track.
  - **Chapter 7** describes the hypergraph-of-entity, introducing it as a general model for entity-oriented search. It covers an initial characterization of the representation model, as well as an evaluation based on the INEX 2009 Wikipedia collection and the TREC 2018 Common Core track for the ad hoc document retrieval task.
  - **Chapter 8** focuses on the characterization of the hypergraph using the tools from network science or, in some cases, proposing analogous approaches more adapted to hypergraphs. We also study temporal statistics, based on the growth of the index, as documents and associated entities are added.
  - **Chapter 9** explores the hypergraph-of-entity as a general model for entity-oriented search, evaluating three of the four retrieval tasks described in Section 1.3.2. We considered the remaining task, related entity finding, to be subsumed by entity list completion. This way, we proved the generality of our (hyper)graph-based model. Experiments were carried over the complete INEX 2009 Wikipedia collection, relying on document profiles. These were created through keyword extraction and thus indirectly filtered linked entities, to reduce index space complexity.

- **Part iv** reflects on the proposed models and the results of the experiments as a whole, considering future directions not only for hypergraph-based models, but also for general models in information retrieval.
  - **Chapter 10** provides a discussion and an overview on the studied and proposed models, considering their limitations and proposing other possible applications.

  - **Chapter 11** concludes with a consideration about the strengths and weaknesses of three large groups of models for entity-oriented search, motivating continued work in graph-based entity-oriented search. We also suggest several future lines of research, identifying the challenges that need to be solved in order to develop a production-ready system that solves information needs in a general way.

- The **Appendix** covers other relevant work, adjacent to entity-oriented, that would not fit the narrative.
  - **Appendix A** provides a reference on PageRank, where we explored several applications and variations, reducing solutions to a power iteration, whenever an alternative version was provided.
  - **Appendix B** covers a parallel line of research where we proposed fatigued random walks, inspired by John von Neumann's reference to the neural fatigue in the brain. We applied fatigued random walks to the graph-based models, proposing a fatigued random walk score over the hypergraph-of-entity, as well as a Fatigued PageRank over a graph.
  - **Appendix C** organizes reference work about classical information retrieval, learning-to-rank, and graph-based models, with application to entity-oriented search, into approach and affected tasks, while briefly summarizing each contribution.
  - **Appendix D** provides a list with the bibliographic references of the scientific publications generated throughout this doctoral work.

### 1.5.3 Contributions

You should read this thesis if you are interested in:

- Gaining perspective towards general models to solve information needs (Chapter 9).
- Modeling retrieval approaches completely based on graphs, e.g., without the need for inverted indexes (Chapters 6, 7 and 9; see also Section 2.2).
- Understanding how hypergraphs can be used for jointly representing terms from documents in corpora, and entities from statements in knowledge bases (Chapters 7).
- Understanding how random walks can be used in a representation-driven retrieval model to rank different units of information (Chapter 9).
- Learning how to tackle the characterization of a hypergraph, by applying or adapting traditional network analysis approaches (Chapter 8).
- Learning how to manipulate PageRank as a random walk based approach for ranking through its different components (Appendix A; see also Section 2.2.8).
- Learning about neuronal fatigue and how it was used to introduce a novel random walk constraint, as well as a Fatigued PageRank (Appendix B).

The main contributions, by order of relevance (as we perceive it), are the following:

1. Hypergraph-of-entity and random walk score (Chapters 7 and 9).
2. Army ANT evaluation framework (Section 5.2).
3. ANT search engine prototype (Section 5.1).
4. Characterization approaches for hypergraphs (Chapter 8).
5. Graph-of-entity and entity weight (Chapter 6).
6. Simple English Wikipedia Link Graph with Clickstream Transitions 2018-12 (Section 4.2.1).

## SUMMARY

In this introductory chapter, we have established context for this thesis, both by providing an historical perspective of information retrieval, as well as the web, knowledge graphs, and real-world networks, and the transition from text search to entity-oriented search. We then presented a motivation for consolidating models, through examples of unified models already explored in other domains. This provided an incentive for general models in information retrieval to be explored, and for unified frameworks, capable of maximizing available information in the process of retrieval, to be proposed. We then introduced combined data and four retrieval tasks central to the creation of a unified model for entity-oriented search. Next, we presented the problem statement, describing the three main goals of this doctoral work, and formalizing them as a thesis statement to be proven. Finally, we described the research line that led us from one step to the next in proving this thesis, presenting the thesis outline over each part and chapter, along with the main reasons to read this thesis, as well as the main contributions that we produced during this doctoral work.

# Part I

# STATE OF THE ART

# 2 GRAPH-BASED ENTITY-ORIENTED SEARCH

## Contents



Only recently has entity-oriented search been conveniently defined and described as an area [62]. While several graph-based approaches have been generally used in information retrieval, graph-based entity-oriented search is still in its infancy. It lies within this area the ability to tackle issues like the combination of heterogeneous information sources or the generalization of entity-oriented search tasks — all available information, structured or unstructured, should be available for cross-referencing, collectively contributing to solving the users' information needs. Likewise, individual tasks leading to the answer might benefit from a departure from modularity and into a more intertwined approach, where intermediate computations from any task should be able to contribute to other tasks, seamlessly at any step. In order to develop such a holistic approach to entity-oriented search, we must first compile a comprehensive guide with a high-level view over information retrieval, and in particular the developments leading to entity-oriented search and the overall usage of graphs in the area. Our goal with this survey was to solve for this need, striving to be complete in the sense of coverage, as opposed to being exhaustive, and showing the potential for tackling information retrieval as the analysis of a complex network.

Entity-oriented search tasks heavily rely on exploiting unstructured and structured collections. Moreover, it is frequent for text corpora and knowledge bases to provide complementary views on a common topic. While, traditionally, the

retrieval unit was the document, modern search engines have evolved to also retrieve entities and to provide direct answers to the information needs of the users. Cross-referencing information from heterogeneous sources has become fundamental, however a mismatch still exists between text-based and knowledge-based retrieval approaches. The former does not account for complex relations, while the latter does not properly support keyword-based queries and ranked retrieval. Graphs are a good solution to this problem, since they can be used to represent text, entities and their relations. In this chapter, we examine text-based approaches and how they evolved in order to leverage entities and their relations in the retrieval process. We also cover multiple aspects of graph-based models for entity-oriented search, providing an overview on link analysis and exploring graph-based text representation and retrieval, leveraging knowledge graphs for document or entity retrieval, building entity graphs from text, using graph matching for querying with subgraphs, exploiting hypergraph-based representations, and ranking based on random walks on graphs. We then cover available evaluation benchmarks and datasets for entity-oriented tasks, focusing on conferences, evaluation forums and their tracks. We close with a discussion on the topic and a view of the future to motivate the research of graph-based models for entity-oriented search, particularly of joint representation models for the generalization of retrieval tasks.

The structure of this chapter is organized as follows:

- **Section 2.1** introduces the classical models of information retrieval and how they influenced and led to applications in entity-oriented search [§2.1.1]. It also introduces learning to rank, highlighting entity-oriented search applications [§2.1.2].
- **Section 2.2** focuses on describing graph-based models with strategies applicable to entity-oriented search. We start by introducing classical link analysis [§2.2.1], and text representations as a graph [§2.2.2]. We cover retrieval processes based on knowledge graphs, as well as their construction [§2.2.3]. We then study retrieval strategies based on entity graphs directly built from text [§2.2.4], and explore their tensor-based representation [§2.2.5]. We also cover graph matching, which is an important part of the semantic web, used in SPARQL for querying RDF [§2.2.6]. We cover several hypergraph-based models, used for different representation and retrieval tasks, including unified indexes, modeling complex document structures, or establishing higher-order dependencies to rank documents [§2.2.7]. Finally, we survey random walk based models, focusing on PageRank variations with several concrete applications in entity-oriented search [§2.2.8].
- **Section 2.3** covers contributions with a clear evaluation strategy in entity-oriented search. It also covers benchmarks and datasets, usually provided as test collections by evaluation forums, such as TREC, INEX, and CLEF. Given the general relevance of TREC and INEX for the IR community, we also cover several tracks of both events that are relevant for entity-oriented search, including entity ranking, question answering, or open search tasks.
- **Section 2.4** begins by presenting several observations about the area [§2.4.1]: justifying the need for a state of the art in graph-based entity-oriented search; commenting on the relations between entity-oriented search and semantic search; clarifying the definition of graph-based models, as used throughout this thesis and across the literature; and further motivating the study of hypergraphs and higher-order dependencies in information retrieval, as a succession to graphs' first-order dependencies. We provide an overview on the reviewed strategies for entity-oriented search [§2.4.2], segmenting them by approach and tasks, for classical, learning to rank, and graph-based models. We do an automatic analysis of the bibliography, both based on the BibTeX file

included on this thesis, and on the reviewed literature that we documented in the doctoral wiki [§2.4.3]. We close the chapter by reflecting on the future of graph-based entity-oriented search [§2.4.4].

## 2.1 FROM TEXT-BASED TO ENTITY-ORIENTED SEARCH

Until recently, search has been focused on the retrieval of documents, a unit of retrieval that frequently represents a partial solution to the information needs of the users. This assigns to the users the task of further analyzing documents from a provided ranking, in order to seek the exact answers to their questions. Furthermore, not only are verbose queries increasingly frequent (cf. Gupta and Bendersky [116, §1.2]), but also are entities more frequently mentioned in queries (cf. Bautin and Skiena [4]). Appropriately, entity-oriented search has been gaining relevance as an encompassing area of research [62], with multiple work unknowingly contributing to this larger area, either by focusing on semantic search[1], question answering, hybrid search, object retrieval, entity search, retrieval or ranking, or other generic approaches that leverage entities, such as document retrieval, sentence retrieval, or learning to rank. Moreover, the Karen Spärck Jones Award[2], an important distinction in the area of information retrieval, has been granted to Krisztian Balog, in 2018, greatly due to his referential work on entity-oriented search.

In this survey, we are particularly interested in graph-based models for entity-oriented search. Nonetheless, we consider it fundamental to first understand the basic concepts of information retrieval, in particular regarding the evolution of text-based approaches to accommodate the needs of entity-oriented search. In the following sections, we provide an introduction to classical information retrieval models (Section 2.1.1) and learning to rank models (Section 2.1.2). Each section then leads to applications of entity-oriented approaches based on each class of models.

### 2.1.1 Classical models

Classical information retrieval models broadly include exact match models, the vector space model, and probabilistic models [12]. In exact match models, like the Boolean model, retrieval is done through set operations of union, intersection and negation. However, there is no order to the retrieved documents and thus there is no relevance ranking. On the other side, in the vector space model, documents and queries are represented as vectors of terms, supporting term weighting. This enables documents to be ranked according to their relevance for a particular query. Perhaps the most widely known ranking function for this model is TF-IDF [8, 9], where each Term Frequency (TF) of query terms is measured for each document and multiplied by its Inverse Document Frequency (IDF) [9]. The IDF decreases the influence of terms that are frequently found in the collection and therefore have a low discriminative power. Another important heuristic for ranked retrieval is Pivoted Document Length Normalization (PDLN) [10, 11], which is used as part of term frequency normalization. It mitigates the impact of document length in relevance ranking, without completely discarding information on the length of the original document.

The underlying concepts of term frequency, inverse document frequency and pivoted document length normalization are transversal to most retrieval models. For

1 Semantic search as a task either refers to the semantically informed retrieval of documents, or to the retrieval of entities or relations over RDF graphs. We cover work on either approach, as both tasks are entity-oriented, using semantic search indiscriminately in both cases.

2 Karen Ida Boalth Spärck Jones was an influential information retrieval scientist, responsible for the creation of the inverse document frequency (IDF), one of the three fundamental concepts in information retrieval, the other being term frequency and document length normalization. See https://irsg.bcs.org/ksjaward.php for more information on the award.

instance, in probabilistic models, ranking functions like BM25 [31] also use term frequency, tackling normalization in a different way, but using the same approach for pivoted document length normalization (see the explanation by Rousseau and Vazirgiannis [16, §5.2], based on the work about information retrieval heuristics by Fang et al. [117], and Lv and Zhai [118]). While BM25 can be seen as the probabilistic counterpart of TF-IDF, there are other widely adopted probabilistic models, like language models, divergence from randomness, Bayesian networks or Markov networks. PageRank can also be classified as a probabilistic model, since it can be modeled as a stochastic process, however we cover it in Section 2.2, as a graph-based model.

Language models [35] rank documents based on the probability of the query term given the document is relevant. For multi-keyword queries, probabilities are multiplied. Language models take advantage of smoothing, usually Jelinek-Mercer or Dirichlet, in order to consider documents with missing query terms, or even documents with none of the query terms. A similar smoothing strategy is also explored in PageRank (see Equations A.2 and A.5 in Appendix A). By considering the prior probability of a term, we broaden the notion of relevance, taking into account query-independent evidence — i.e., a document might be relevant solely due to its terms, however it will often be less relevant than documents with a strong query-document relation.

Divergence from randomness [119] is a probabilistic model where we measure the information gain of a term given a document. This model is a generalization of Harter's indexing model [24], where two Poisson distributions (hence 2-Poisson) were combined in analogy to TF and IDF, using the notion of eliteness to describe documents with a more prominent presence of query terms, when compared to other documents. A similar approach is taken in divergence from randomness, but there is a wide range of models to select from, as opposed to only being able to use the Poisson distribution. The gain in divergence from randomness is computed based on the risk of a term not being informative (akin to TF), as well as the probability of the term given the document not being random according to the collection (akin to IDF). The absolute term frequencies are normalized according to the standard document length (akin to document length normalization). Risk is also known as the after effect or the first normalization and it is defined as the probability of a term belonging to a document given the document belongs to an elite set (i.e., documents that are more representative of a given term in the collection). This probability is usually computed based on a divergence from randomness model. Another basic divergence from randomness model is then selected for the IDF-like part of the model, which also defines the normalization approach for TF.

Finally, Bayesian networks and Markov networks are probabilistic graphical models, respectively based on a directed acyclical graph and an undirected graph (potentially with cycles). In particular, the inference network model [29], based on a Bayesian network, models the dependencies between the information need, the query, the representation nodes (e.g., terms, phrases, etc.) and the documents. A similar approach, perhaps less classical, can be taken based on Markov networks, by modeling the dependencies between query terms and documents [120]. The advantage is that terms can be modeled in a more expressive manner, for instance as fully independent, sequentially dependent or fully dependent, where there is no clear direction, but rather bidirectional dependencies based on adjacency.

#### *Applications to entity-oriented search*

Some of the first approaches to entity-oriented search revolved around classical retrieval models, through the reuse of well-established text-based ranking techniques, as presented above. They include, most notably, defining virtual documents to represent entity profiles, or integrating results obtained from an inverted index and a triplestore.

Bautin and Skiena [4] presented what they considered to be the "first-in-literature" implementation of an entity search engine. Their first step was to find evidence that the task was relevant, based on the analysis of the AOL dataset, with 36 million web search queries. They found that 18–39% queries directly referenced entities and 73–87% contained at least one entity. They then proposed a concordance-based model for entity representation, along with an adaptation of Apache Lucene's[1] TF-IDF scoring scheme. Each concordance[2] (a virtual document) was built from the concatenation of all sentences containing the entity it represented, optionally for a given period of time (e.g., a month). Appropriately, they also proposed a time-dependent scoring function, modeling user interest in an entity as a function of time, and optimizing parameters based on the frequency of entities in the AOL query log. Finally, experiments were run over the entities extracted from an 18 GiB collection of US news articles, collected through the Lydia pipeline [121]. They proposed a method for evaluating entity search by comparing the results list with the corresponding list obtained through a juxtaposition score [121]. The juxtaposition score measures the upper bound of the probability of two entities occurring in the same sentence under the assumption of independence. By obtaining the results list from Lucene and the results list based on the top related entities according to juxtaposition, the lists were then compared using the $K_{min}$ distance from Fagin et al. [122], showing the best results for phrase queries with the slop parameter (word-based edit distance) equal to the number of query terms.

Bhagdev et al. [123] presented an example of hybrid search, where they combined keyword-based search with semantic search, showing that their approach outperformed either of the alternatives when individually used. They indexed text documents using Apache Solr[3]; they stored annotations generated by an information extraction system on a Sesame triplestore[4]; and they linked the extracted relations by annotating the provenance of the triples with the document of origin. At retrieval time, this enabled them to do keyword search over the inverted index, metadata search over the triplestore using SPARQL, and keywords-in-context search by retrieving text documents and matching them with triples through the provenance annotation. Their evaluation was based on 21 queries over a collection of 18 thousand technical documents. When comparing keyword search with metadata search, they obtained the best recall for keyword search (0.57 versus 0.40) and the best precision for metadata search (0.85 versus 0.56). However, when combining both approaches in a hybrid search, they obtained the best overall result, with a precision of 0.85 and a recall of 0.83. While the authors did not specifically mention it, this is clearly an example of entity-oriented search over combined data.

Pound et al. [3] proposed a formal model for ad hoc entity retrieval, but they used the designation *object* instead of *entity*, in the context of the web of data (the semantic web). They defined the task based on a keyword query for input, with an identifiable query type and query intent. The query was then processed over a data graph, returning a ranked list of resource identifiers (entities). Based on the analysis of real query logs from a commercial search engine, they also proposed five query categories for ad hoc entity retrieval: entity query, type query, attribute query, relation query, and other keyword query. These query categories can be mapped into specific tasks of entity-oriented search [62]. For instance, an entity or type query could be solved through ad hoc entity retrieval over virtual documents [4, 124], while an attribute or relation query might be solved through related entity finding or entity list completion, if attributes were indexed as entities. They also discussed result presentation, proposing that each retrieved entity should be decorated with linked entities, in order to explain or contextualize the result. For evaluation, they established a baseline ranking approach based on TF-IDF for RDF, over an inverted

1 http://lucene.apache.org
2 A concordance is a list of terms and their context. In this case, the concordance is about entities and their context.
3 http://lucene.apache.org/solr/
4 Sesame is now known as Eclipse RDF4J: http://rdf4j.org/.

index. They then compared it with two other approaches based on the reranking of the top-5 results given by the baseline: a random reordering and an ideal reordering based on the assessments of human judges. Assessment was done in regard to the query resource (indicative of intent) and in regard to the full text query, both evaluated by the human assessors. In both cases, the baseline was able to surpass its randomly reranked version, but, as expected, it was still below the ideal reranking. They also evaluated the stability of the metrics, using bootstrapping to generate one million samples over the 264 assessed queries, and then compared the mean of each metric over all samples with the empirical mean over the original data, thus verifying that they approximated each other.

Koumenides and Shadbolt [125] proposed a Bayesian inference model for entity search. They combined link-based and content-based information as defined through RDF object properties and data properties. A query network was defined based on entity and property evidence, that could either be provided explicitly as entities or implicitly as a combination of keywords. Common object or data properties were modeled through common identifier nodes $O_i$ and $D_j$. By keeping separate nodes $o_{k,i}$ and $d_{k,j}$ for different instances of object and data properties, the model was able to use query nodes as evidence of object property identifiers, as well as data property identifiers or instances. This could then be further expanded into entities, or terms in the literal space. Unfortunately, the authors did not provide appropriate evaluation of their approach, making it unclear how it performs in relation to other approaches.

Urbain [126] presented a pipeline for entity-oriented sentence retrieval, proposing a strategy for the integration of terms (context), entities and their relations. He used a Markov network for modeling the dependencies between a pair of entities, a relation and a context, using a fully connected approach. No external knowledge bases were used. Instead, sentences in the form of triples $\langle \textit{entity}, \textit{:relation}, \textit{entity} \rangle$ were obtained through natural language processing, extracting structure from documents and natural language queries. This enabled the construction of a Markov network that, together with user relevance feedback, was able to rank sentences by leveraging entities and relations. He compared several models, based on different combinations of feature functions for the Markov network. This included dependencies between entities, relations, and sentence and document terms. They consistently obtained better results for the proposed entity-relation model, supporting the importance of the entity graph in retrieval tasks.

Raviv et al. [124] proposed a general model for entity ranking, based on a Markov network for modeling the dependencies between the query and the entity. In particular, the model captured the dependencies between: (i) the entity document (i.e., a virtual document) and the query; (ii) the entity type and the query target type; (iii) the entity name and the query. A profile based approach, supported on a Dirichlet smoothed language model, was used for scoring entity documents. A filtering approach, based on the Kullback-Leibler divergence between the probability distributions of the entity and query types, was used for scoring the entity type. The entity name was scored using a voting or a global approach. The voting approach was based on the language models of retrieved entity documents relevant to the query. The global approach was based on the pointwise mutual information between the entity name and a query term. Evaluation was done over the INEX 2006 and 2009 Wikipedia collections, based on the topics and relevance judgments from the Ad Hoc track. They experimented with three entity ranking models based on fully independent, sequentially dependent and fully dependent query terms, that combined entity document, entity type and entity name dependency models. In 2007, they obtained the best results, according to MAP, using full dependence over a ranking function based on the combination of the three dependency models. In 2008 and 2009, they obtained the best results, according to infMAP [127, §2.5], using sequential dependence for the same ranking function.

Raviv et al. [73] also tested the cluster hypothesis for entity-oriented search, i.e., the hypothesis that "closely associated entities tend to be relevant to the same requests". They experimented with four similarity metrics: (i) an exponential function of the shortest distance between any two categories of a pair of entities in the Wikipedia's category graph (*Tree*); (ii) the cosine similarity between the binary category vectors of the two entities (*SharedCat*); (iii) an exponential function of the negative cross entropy between the Dirichlet-smoothed unigram language model for the documents resulting from the concatenation of all the Wikipedia articles for each category (*CE*); and (iv) the cosine similarity between two vectors obtained from explicit semantic analysis (*ESA*). For each similarity measure, three different weighting schemes were used: $L_{Doc}$, $L_{Doc;Type}$ and $L_{Doc;Type;Name}$. For $L_{Doc}$, the Wikipedia document corresponding to each entity was indexed and directly used to retrieve the entity. For $L_{Doc;Type}$, the similarity between the category set of each entity and the query target type was also taken into consideration. Finally, for $L_{Doc;Type;Name}$, the proximity between the query terms and the entity name was also taken into consideration. Evaluation was carried over the datasets for the 2007, 2008 and 2009 INEX Entity Ranking tracks, which used the English Wikipedia from 2006 and 2008. The authors found that the nearest neighbor cluster hypothesis holds. While result lists frequently contained 10-25% relevant entities, nearest neighbor entities of a relevant entity contained 30-53% relevant entities. Best results were achieved when using the *Tree* and *SharedCat* inter-entity similarity measures and were particularly good for the *Oracle* method, which employed cluster-based reranking based on the true percentage of relevant entities contained in each cluster. Other approaches included the *MeanScore* and *RegMeanScore*, which instead used the average score within a cluster of entities, optionally with regularization.

Bron et al. [128] tackled the task of entity list completion, where, given a textual description for a relation and a given set of example entities, the goal was to retrieve similar entities that respected the specified relation. Supported on language models, they experimented with text-based and structure-based approaches, as well as a combination of both. The text-based approach took advantage of the textual description of the relation, while the structure-based approach used the set of example entities provided as relevance feedback. For integrating both approaches, they experimented with a linear combination, as well as a switch method. The switch method was based on a performance overlap threshold, used to determine whether there was a relevant difference in performance between the two methods. In that case, they selected the method that achieved the highest average precision. Otherwise, when no relevant difference in performance was found, they simply relied on the linear combination. Their experiments showed that both approaches were effective, despite returning different results. They also found that the combination of the two approaches outperformed either one of them when independently used. This further supports the need for a hybrid approach that combines both the strengths of text-based and structure-based features.

Bast and Buchhold [74] presented a novel index data structure for efficient semantic full-text search. They argued that neither classic inverted indexes nor triplestores could handle the problem individually. None of the approaches was able to provide multiple integration steps for different stages of query processing. They exemplified with a friendship relation that could only be found in the text, but should influence retrieved triples, potentially by establishing new connections. This was, however, unsupported by current approaches. Accordingly, they proposed a joint index for ontologies and text. As opposed to traditional keyword queries, they used trees as queries, based on the graphical interface provided by the Broccoli semantic search engine [129]. In order to provide a search mechanism over a tree query, the index distinguished between two types of lists: lists containing text postings, which they called context lists, and lists containing data from ontology relations. Each context list was associated with a word prefix and contained one index item per occurrence of a word starting with that prefix. An index item, in turn, contained the context

ID, a unique ID mapping to either a word or an entity, a score for each word/entity, and the position within the context for each word/entity. Each ontology relation (e.g., *:born-on-date*) was represented by a separate index, containing a list of items, each with a unique ID for a source entity (e.g., *Neil Armstrong*), a unique ID for a target entity (e.g., *August 5, 1930*), and a score for the relation, possibly left as a placeholder for future weighted relations, since it was fixed at 1 for all relations. Query processing was then based on resolving *variable* nodes, *ontology* arcs, *occurs-with* arcs, *has-occurrence-of* arcs, *occurs-in* arcs and *in-range* arcs, by traversing the query tree in a bottom-up recursive fashion. Tree nodes could then be processed based on the context lists and the ontology relation lists of the joint index. They evaluated efficiency, by comparing the inverted index and the triplestore baselines with two approaches (*Map*, linking context ID to entity postings, and *CL*, context lists with word and entity postings) based on their joint index. While the joint index supported all defined queries, these were only partially supported by each baseline individually, but completely supported by both when collectively considered. Overall, they found the joint index approaches to require less disk space, taking similar or less time to query than the baselines.

Zhou [130] wrote a doctoral thesis on entity-oriented search, exploring the topic by distinguishing between querying by entities and querying for entities. In querying by entities, entities were taken as input, while results could either be documents or entities. In querying for entities, entities were returned as output, while queries could either be keywords or entities. He also highlighted the particular case of querying by and for entities, where entities were both taken as input and output. For querying by entities, he presented contributions on entity-centric document filtering. He proposed using an entity page, such as the associated Wikipedia page, to describe an entity in the query. This is different from the virtual document approach, described in previously covered work [4, 124], in the sense that it is the entities in the query that are represented as documents, as opposed to the entities in the index. Regarding querying for entities, they proposed a content query language (CQL) over a relational-model based framework, as a solution to a data-oriented content query system. As opposed to keyword or entity queries, this querying approach required advanced technical knowledge, similar to SQL or SPARQL. In order to support CQL, they used an advanced index layer that included a joint index and a contextual index. The joint index combined pairs of keywords, keyword and data type, and pairs of data types, storing, for each occurrence, the document identifier, the position of the first keyword or data type and the distance to the second keyword or data type — only keywords or data types within a distance were considered for indexing. The contextual index combined ordered pairs of keywords, keyword and data type, data type and keyword, and pairs of data types, storing, for each occurrence, the document identifier, the position of the source keyword or data type and a list of tuples, each with an offset and term occurring within the given distance of the source. Regarding querying by entities and for entities, they proposed a relational entity search framework based on a target relation $\mathtt{r}(\mathtt{q}, \#\mathtt{E})$, where $\mathtt{r}$ was the relation, $\mathtt{q}$ was the query and $\#\mathtt{E}$ was the entity type that should be returned by the query (e.g., *FounderOf('microsoft', #person)*). In order to deal with unseen cases, they proposed a distantly supervised ranking method to learn a relation-specific ranking function for entity search. Based on the book on entity-oriented search by Balog [62], we can reclassify the tasks explored by Zhou in his doctoral thesis. The task of querying by entities, where results are documents, can be classified as ad hoc document retrieval leveraging entities [62, Ch.8]. The task of querying for entities, where the query is keyword-based, can be classified as ad hoc entity retrieval [62, §3.1]. Finally, the task of querying by entities and for entities, where both the query and the results are entities, can be classified as related entity finding [62, §4.4.3].

Dietz and Schuhmacher [131] introduced Queripidia, as a set of knowledge portfolios. A knowledge portfolio represented a query-specific collection of relevant entities, combined with text passages from the web that explain why the entity is rel-

evant to the query. They used two main datasets in order to develop a working prototype: the FACC1 entity link collection[1], a Freebase annotation of the ClueWeb corpora, automatically generated by Google; and the ClueWeb12[2], Category A dataset, used in the TREC Web track, where several test queries were also provided. Besides text passages, neighboring entities from the knowledge base were also included in the explanation, in order to provide additional context. In turn, each neighboring entity was associated with its own explanation in the context of the same query. This work is further detailed in Dietz et al. [132], where they explored several entity ranking approaches in order to understand whether the combination of documents and a knowledge base would improve entity ranking. All approaches were based on language models. They explored two different entity profile approaches: (i) using textual evidence surrounding the entity to establish context, and (ii) using the entity's Wikipedia page to represent the entity. Ranking based on the entity context involved document retrieval using the sequential dependence model by Metzler and Croft [120], followed by entity context construction and reranking based on the entity profile. Ranking based on Wikipedia was simply modeled as document retrieval, using the entity's page as its profile. The best retrieval performance was obtained based on the entity context, particularly for a window size of 50 words, when compared to the Wikipedia based approach. However, the best overall performance was achieved using a rank fusion technique based on the two methods, showing that the combination of text and knowledge in fact outperforms each individual approach.

**REMARKS** The clear advantage of reusing classical information retrieval models for entity-oriented search is the support on well-established approaches that have been researched for text-based applications over the years. On the other hand, it is quite limiting to discard or flatten complex entity relations that can't be represented using an inverted index, but would otherwise be helpful to solve an information need.

### 2.1.2 Learning-to-rank models

Learning-to-rank [133, 134] provides several approaches for applying existing machine learning techniques to document or entity ranking. A ranking function is learned from a training set based on a matrix X of document features — which can be query-dependent, for multiple queries, or query-independent — along with a vector Y with the relevance grades — that measure the matching degree between a document and a query. Query-dependent evidence includes features like TF, IDF, document length, BM25, or language model based weighting, while query-independent evidence includes features like the number of slashes in the URL and the length of the URL, or link analysis metrics like indegree and outdegree, HITS authority and hub scores, or PageRank. Depending on the learning-to-rank approach, the actual Y variable might be optimized over different loss functions. In particular, there are three main approaches to measuring the loss when training a model: pointwise, pairwise and listwise. When using pointwise loss, each individual document is considered only in regard to the query, based on the computed features that already account for this relation. It measures the distance between the ranking function to be learned and the relevance grade. When using pairwise loss, the relative ranking of pairs of documents is instead considered. This requires using the relevance grade to compare the two documents. In practice this usually means that we instead consider the combined features of the two documents and that the loss function becomes an indicator of relative order. Finally, when listwise loss is considered, performance metrics like NDCG are instead used to compute the loss over a complete ranking, instead of individual or pairs of documents. Out of the

1 http://lemurproject.org/clueweb09/FACC1/
2 http://lemurproject.org/clueweb12/

three loss functions, listwise loss is the only one that takes into account the group structure of the rankings.

Ai et al. [38] have drawn the attention to the fact that, despite loss being computed as pointwise, pairwise or listwise, the learned ranking function is still pointwise in regard to a document. This follows the classical approach of information retrieval, where ranking is based on $score(q, d)$, be it TF-IDF, BM25 or language models. The authors proposed GSF (Groupwise Scoring Function) to take advantage of relations between multiple documents during ranking, separating them into groups of size $m$. They presented this approach, based on deep neural networks, as a generalization of other existing approaches for pointwise scoring. The neural network architecture they proposed is invariant to input document order and limited to comparisons between small groups of documents. Each group is built from the permutations of size $m$ over $n = |D|$ documents. This means that, for groups of size $m = 1$ over $n = 2$ documents, this is equivalent to pointwise scoring and pairwise loss, where one deep neural network is trained per document pair. It also means that, for groups of size $m = 1$ over $n$ documents, where $n$ is the size of the list, this is equivalent to pointwise scoring with listwise loss. Evaluation was done over the MSLR-WEB30K dataset[1], using NDCG@5 to compare RankNet [135], MART and LambdaMART [136], GSF and LambdaMART + GSF. GSF was able to outperform RankNet, but not MART or LambdaMART. However, the best results were obtained for the combination of LambdaMART + GSF. The GSF model also obtained better results for higher group sizes $m$, showing that inter-document comparisons are important in discriminating relevance. This is also a strength of graph-based models like PageRank or HITS, which have been focused on the query-independent ranking of documents through their relations. In a sense, this also aligns with the cluster hypothesis, where documents similar to relevant documents are probably also relevant.

#### *Applications to entity-oriented search*

Chen et al. [137] explored the task of answer sentence retrieval, where sentences were ranked in respect to an input question. The challenge was that the best results did not necessarily contain the terms of the query, resulting in a lexical mismatch between the sentences and the question. This was an indicator that semantic features could be useful in tackling the problem. The authors proposed a learning to rank approach, establishing a baseline supported on Metzler-Kanungo (MK) features [138] — sentence length, sentence location, exact match of query in sentence, term overlap of query terms in sentence, synonym overlap of query terms in sentence, and language model (i.e., likelihood of query terms being generated by the sentence language model). They then proposed and tested two new semantic features, one based on ESA (Explicit Semantic Analysis) [139] (the cosine similarity between the query and sentence ESA vectors), and another one based on the word2vec skip-gram approach [140] (the average cosine similarity between any query-word vector and any sentence-word vector). Through the evaluation of three learning-to-rank approaches — linear regression, coordinate ascent, and MART — they showed that results could be improved by leveraging semantic features. For each approach, they compared four feature configurations: (i) MK; (ii) MK + ESA, (iii) MK + word2vec and (iv) all features. The best results were consistently obtained for all features combined, except for MART, where MK + ESA obtained the best results, despite being closely followed by all features combined.

Lin et al. [141] tackled the task of related entity finding in TREC 2011 Entity track [142], where the goal was to rank the homepages of target entities, given a source entity, a target entity type and a narrative describing the relation between the source and target entities. Their approach consisted of document retrieval (using Yahoo!), entity extraction (using StanfordNER), feature extraction and entity

1 https://www.microsoft.com/en-us/research/project/mslr/

ranking. For document retrieval, the goal was to obtain the homepage of an entity — their best approach was based on querying using the narrative to describe the relation. For entity ranking, they used a learning to rank approach based on features that considered frequency, density, proximity, semantic similarity, and the average rank of web pages, in regard to a candidate entity (e.g., total frequency of the entity in search results, similarity between the query and the entity type). They trained three SVM, one with default hyperparameters, another one with tuned hyperparameters, and a final one after applying feature selection. They discovered that the SVM with tuned hyperparameters performed better than the one with the default hyperparameters, and that the SVM with the selected features performed worse than the tuned SVM. Interestingly, they also discovered that directly using one of their proximity-based features yielded better results by itself. Based on the number of retrieved documents multiplied by the cumulative distance between the query and the entities in the documents, the authors were able to achieve better results than the SVM models. They also compared the tuned SVM with an approach based on a linear combination of all features, obtaining better results for the linear combination, thus finding that their assumption that the SVM would perform better was wrong.

Schuhmacher et al. [143] used a learning-to-rank approach for entity ranking, combining features about documents, entity mentions and knowledge base entities. They experimented with pairwise loss based on a support-vector machine, minimizing the number of discordant pairs in Kendall rank correlation coefficient. They also experimented with listwise loss based on coordinate ascent, minimizing both MAP and NDCG. Several features were considered, based on an initial set of retrieved documents. In particular, they covered features like mention frequency, query-mention similarities, query-entity direct matching and path similarity over DBpedia, query term presence in the entity's Wikipedia article (based on a boolean retrieval model), the retrieval score for Wikipedia pages representing an entity (based on a sequential dependence model with Dirichlet smoothing), the PageRank of the entity's Wikipedia page, and entity-entity features measuring the path similarity between all considered entities (introduced in the model via a semantic smoothing kernel). Evaluation was carried over the REWQ datasets[1], created by the authors over the TREC Robust 2004 dataset and the ClueWeb12 corpus. They compared three baseline and three learning to rank models. The baseline models included the sequential dependence model, the mention frequency, and the PageRank. The learning to rank models included coordinate ascent and two SVMs, with and without a semantic kernel based on the relations between entities. They obtained the best overall results for the coordinate ascent approach. For the REWQ Robust dataset, the best performing individual feature was the sequential dependence model, while, for the REWQ ClueWeb12 dataset, it was the mention frequency. Both resulted in NDCG scores close to the learning to rank models.

Chen et al. [144] studied the effectiveness of learning to rank for ad hoc entity retrieval. They represented an entity based on a document with five fields derived from RDF triples: names, attributes (excluding the name), categories, related entity names and similar entity names (aliases). They then extracted query-entity features based on a language model, BM25, coordinate match, cosine similarity, a sequential dependence model (SDM) and a fielded sequential dependence model (FSDM). This resulted in a total of 26 features (five dimensions per feature, except for FSDM, which resulted in only one dimension). They experimented with a pairwise method (RankSVM) and a listwise method (coordinate ascent, optimized for MAP), comparing with the FSDM baseline, as well as a sequential dependence model and a mixture of language models, both optimized using coordinate ascent (SDM-CA and MLM-CA). They consistently obtained the best results for the two learning-to-rank approaches over test collections from well-known evaluation forums (SemSearch ES, ListSearch, INEX-LD and QALD-2). They also measured the influence of the fields

1 http://mschuhma.github.io/rewq/

and feature groups in the RankSVM approach, overall finding that the related entity names was frequently an important field, and that the SDM related features were in general the most influential.

Gysel et al. [86] have tackled the problem of product search based on representation learning. They proposed the latent semantic entities (LSE) for jointly learning the representations of words ($W_v$), entities ($W_e$), and a mapping between the two ($W$). A string, be it an n-gram from a document or a keyword query, is mapped to the entity space based on the following steps. Given a word represented by its one-hot vector, a learned matrix $W_v$ of word embeddings is used to map the averaged one-hot vectors of the string to its embedding. A word embedding is then mapped to the entity space using a learned matrix $W$ and bias vector $b$ and applying the *tanh* function. An entity can also be represented in the same space, based on its embedding, as defined in the entity embeddings matrix $W_e$. Learning is done based on gradient descent over a loss function $L(W_v, W_e, W, b)$. They evaluated the effectiveness of LSE in an entity retrieval set based on a learning-to-rank pairwise approach (RankSVM), exploring query-independent features (QI), a query-likelihood language model (QLM), and the latent semantic entity representation (LSE). Their best results were consistently obtained for QI + QLM + LSE, tested over different product categories, when compared to QI, QI + QLM, and QI + LSE.

**REMARKS** With learning to rank, we require prior data for multiple queries and the corresponding lists of ranked documents or entities, with associated relevance judgements. This enables the training of a model based on a set of features that characterize the document or entity and the query-document or query-entity relations. A ranking function is learned by fitting these features, while optimizing for the relevance judgments using pointwise, pairwise, or listwise loss functions. Learning to rank inherently supports the integration of signals from heterogeneous sources. It might also contribute to the generalization of entity-oriented search, through multi-task learning, which relies on a shared representation [145, §1.1] or common input-output space [146, §3] to train tasks in parallel. To our knowledge, this hasn't yet been explored for generalizing entity-oriented search tasks. Furthermore, a relevant question is whether machine learning models are able to provide adequate explainability and transparency. Maybe graphs have the edge there? Regardless, they are both worth studying as general solutions to information retrieval. In this work, we focus on graphs, so let us discuss graphs.

## 2.2 GRAPH-BASED MODELS

Search is based on a simple principle developed in the library. In order to find a relevant page of a book, based on a given keyword, we originally had to scan the book, page by page. This was a time consuming task, particularly for books with a large number of pages. The problem was solved through the back-of-the-book index, where a list of manually selected keywords would point to the pages mentioning a given concept. Taking only a few pages and using an alphabetical order, this approach was more efficient than reading the whole book. The same principle applies when indexing a collection of documents in a computer. A collection that would take a long time to be fully scanned is condensed in an inverted index, where terms point to lists of documents, storing statistics like the frequency or the positions of the term in the document. As opposed to the back-of-the-book index, an inverted index contains most of the terms in the collection, usually discarding frequent words (stopwords) and sometimes storing a reduced form of the word (obtained from stemming or lemmatization). Automatization means that a larger volume of data can be processed efficiently, and stored statistics can be used as a way to measure relevance. However, one thing that is lost with the inverted index is the ability to relate concepts. In the back-of-the-book index, a domain expert might

provide associations between concepts (e.g., using 'see also') or use keywords that are not explicitly mentioned in the page despite being more adequate for search. The inverted index is usually focused on representing the document as is, however we can use techniques like query expansion or latent semantic indexing to establish new connections that make documents more findable. With query expansion we can, for instance, also consider the synonyms of the query keywords to increase recall. With latent semantic indexing we can establish new relations based on contextual similarity, or we can use approaches like word2vec or explicit semantic analysis for a similar purpose.

Another relevant source of concept relations are knowledge bases, which are more explicit and can be used to improve retrieval by leveraging the semantics provided by entities. Due to the complex relations between entities, knowledge bases are usually represented as graphs. The most frequently used model for this is RDF (Resource Description Framework), a tripartite labeled directed multigraph. In an RDF graph, each relation is modeled by three linked nodes known as a triple — a subject (entity), a property (relation), and an object (entity or attribute). Other approaches include topics maps or the property graph model. Topic maps model topics through their associations and occurrences. Topics are analogous to keywords in the back-of-the-book index, while occurrences are analogous to the page numbers. Associations can represent $n$-ary connections between topics, similar to the role of the 'see also' expression in the back-of-the-book index. In the property graph model, relations are captured between entities, but properties are not explicitly a part of the graph, being externally associated with nodes and edges instead. In comparison to RDF, attributes and relations are not represented as nodes in the graph, but are instead stored in a node property index and defined as edge labels, respectively. RDF is a strong model for inference, while the property graph model provides a solid base for ranking entities without having to consider the effect of tripartite relations or having to compute a projection over one the three modes. Knowledge graphs [62, §1.4.4] are usually queried through a structured language like SPARQL, used for graph matching. Unlike unstructured keyword-based queries, SPARQL is not user-friendly, in the sense that it requires a certain degree of technical expertise that is more distant from natural language. There is a need for keyword-based retrieval over knowledge graphs, but also for the structured data that knowledge graphs usefully provide to improve the effectiveness of document retrieval. Furthermore, understanding graph-based models for representing, retrieving or otherwise manipulating text and/or knowledge is an essential step towards approximating a more general solution to information retrieval. On one side, graphs are ideal for dealing with the problem of heterogeneity [147]. On the other side, and perhaps more importantly, awareness about a diverse set of graph-based models, from multiple application contexts, is essential to support the quest for finding a joint representation model of terms, entities and their relations, along with a universal ranking function that can be used for entity-oriented search and, eventually, for information retrieval in general.

Many of the graph-based techniques currently applied to entity-oriented search, were surveyed in 2005 by Getoor and Diehl [148], who grouped them into the area of link mining[1]. They covered tasks from link analysis, community detection, entity linking, and link prediction that, in some way, provide a workbench for developing graph-based entity-oriented search. In this section, we survey the usage of graph-based models for multiple retrieval tasks, from modeling documents as graphs, to providing query-dependent and query-independent evidence of document or entity relevance. In Section 2.2.1, we present classical link analysis approaches, covering PageRank, HITS and heat kernel. In Section 2.2.2, we introduce graph-based representations of documents, used for ad hoc document retrieval. In Section 2.2.3, we present retrieval methods based on knowledge graphs, for improving or aug-

1 There is not much evidence of link mining as an area beyond this survey, which leads us to believe that, albeit a good one, this showed no relevant adoption by the community.

menting document retrieval, as well as for entity retrieval. In Section 2.2.4, we explore approaches that rely on entity graphs built directly from text corpora, and in Section 2.2.5 we cover tensor based approaches for representing entity graphs. In Section 2.2.7, we provide an overview on hypergraph-based models, covering tangential work with applications to entity-oriented search. Finally, in Section 2.2.8, we focus on random walk based models, in particular covering applications of PageRank to entity-oriented related tasks.

### 2.2.1 Link analysis

Classical graph-based models in information retrieval include HITS and PageRank, two link analysis algorithms developed to rank pages in the web graph. In 1999, Kleinberg [149] proposed the hypertext induced topic selection algorithm (HITS) as a combination of an authority score, based on incoming links, and a hub score, based on outgoing links. The computation of HITS is frequently done over a query-dependent graph, built from a root set of pages that are relevant to the query. The root set can be retrieved using a classical model like TF-IDF or BM25 and it is then expanded into a base set that includes all outgoing links and a subset of incoming links. While the number of outgoing links is usually small, the number of incoming links can be too high for an efficient computation. Thus, a parameter $d$ is used to define a ceiling for the number of incoming links to consider. When the number of incoming links surpasses $d$, then only a random sample of size $d$ is considered, otherwise all incoming links are considered. In its query-dependent application, HITS is more expensive than PageRank for ranking, since it cannot be computed offline. Like PageRank, HITS is also related to the leading eigenvector of a matrix derived from the adjacency matrix. Interestingly, the authority and hub scores are related to the leading eigenvectors of $AA^T$ and $A^TA$, respectively, both sharing the same eigenvalue [150, §3.2].

Also in 1999, Page and Brin [113] proposed PageRank as a way to measure the importance of web pages. PageRank [151] is an elegant algorithm that offers multiple interpretations and computation approaches. It can be seen as the solution to a linear system [152, 153], or as the eigenvector of the Markov chain derived from the graph — after adding a teleportation term to the transition probabilities, in order to deal with sinks (i.e., pages without any links to other pages). It can be solved through Gaussian elimination, power iteration or even Monte Carlo methods [154]. Conceptually, PageRank is a random surfer model, where the probability of visiting a node reflects the behavior of a user that is randomly navigating the web by clicking hyperlinks, while occasionally jumping to a new page. This model is recursive, in the sense that it results in a centrality metric where the importance of a node depends on the importance of its neighbors — the better connected a node is, both through quantity (i.e., many nodes) and quality (i.e., nodes that are themselves well connected), the higher the PageRank. Research about PageRank has led to many applications [155], exploring contextual information (e.g., Topic-Sensitive PageRank [156]), combinations of features (e.g., Weighted PageRank [157]), alternative smoothing approaches (e.g., Dirichlet PageRank [158]) or historical evidence (e.g., Multilinear PageRank [114]). One of the variants, Reverse PageRank [159], consists of simply reversing the edge direction and computing PageRank for this complementary graph. It is to PageRank what the hub score is to the authority score in HITS. Bar-Yossef and Mashiach [160] have shown that the Reverse PageRank is not only useful to select good seeds for TrustRank [161] and for web crawling, but also, more interestingly, for capturing the semantic relatedness between concepts in a taxonomy. According to Gleich [155, §3.2], Reverse PageRank can be used to determine *why* a node is important, as opposed to simply identifying *which* nodes are important, something that PageRank already solves. The success of PageRank in complementing itself through different applications is a sign of the usefulness of

random walks in solving diverse tasks, which is a useful characteristic in the design of general models.

Node importance is generally measured based on the number of incoming links (as we have seen with HITS authority and PageRank) or based on the favorable structural position of a node (e.g., closeness [39], betweenness [44]). Besides node importance, node relatedness can also be measured as a type of structural similarity, usually based on whether two nodes share links to or from a common node. Van and Beigbeder [162] explored the effect of node relatedness in the retrieval of scientific papers based on a user profile. They experimented with bibliographic coupling and co-citation as reranking strategies. In bibliographic coupling, two papers are related if they cite a common publication. In co-citation, two papers are related if they are cited by a common publication. For measuring co-citation, they implicitly built a graph based on Google search results for pairs of paper titles, as well as based on data from the Web of Science. Based on the 20 content-only topics from INEX 2005, each representing an information need of a user, the authors selected approximately five papers per topic to establish a user profile. Using Zettair[1], they then indexed the collection of papers, ignoring those used to build user profiles. They retrieved 300 papers for the 20 topics, based on Dirichlet-smoothed language models, and used this as the baseline. Results were then reranked based on bibliographic coupling, co-citation using the Web of Science, and co-citation using Google. They obtained a consistent improvement over the baseline only for the Google-based co-citation reranking (P@10 increased from 0.62 to 0.68).

Link analysis can also be approached through kernels, supporting both the measurement of importance and relatedness. Ito et al. [163] explored von Neumann kernels as a unified framework for measuring importance and relatedness, using different parameter configurations to go from co-citation or bibliographic coupling ($n = 1$) to HITS (large values of $n$). They also identified two limitations of co-citation relatedness: (i) two nodes are considered to be related only when they are cited by a common node; (ii) relatedness only takes into account the number of nodes commonly citing two nodes, as opposed to also considering the differences in popularity of the two nodes (e.g., co-citing a generic web site and Google might not be an indicator of relatedness, given the popularity of Google). As a solution, they proposed the use of Laplacian and heat kernels, which enabled them to control the bias between relatedness and importance, while effectively mitigating the identified limitations.

The heat kernel has also been studied as a type of PageRank [164, 165], establishing an analogy with PageRank's alternative notation [166, §1.5], frequently used for personalization tasks. This is shown in Equation 2.1, for a personalization vector $E$, the complement of the damping factor $\beta = 1 - d$, and the Markov matrix $\mathcal{M}$ representing the graph.

$$PR = \beta \sum_{n=0}^{\infty} (1-\beta)^n E\mathcal{M}^n \tag{2.1}$$

Converting to this notation requires knowledge of linear algebra and, in particular, of the Neumann series [167, Eq.6] for solving matrix inversions. As an analogy, Chung [164] proposed the heat kernel PageRank, illustrated in Equation 2.2.

$$HKPR = e^{-t} \sum_{n=0}^{\infty} \frac{t^n}{n!} E\mathcal{M}^n \tag{2.2}$$

Comparing both equations, we find that $\beta$ is analogous to $e^{-t}$ and, while $1-\beta \neq \frac{t^n}{n!}$, there is still an approximated correlation of 85% between both terms (calculated for

1 http://www.seg.rmit.edu.au/zettair/

$n \in [0, 150]$ using 0.001 increments). Chung found the heat kernel PageRank to have graph partitioning applications. This is interesting in the sense that PageRank can both be used to measure node importance if applied globally, as well as to cluster nodes if applied locally. Relevance can be modeled as node importance (e.g., using PageRank as query-independent evidence, or HITS authority as query-dependent evidence), but we also know that the cluster hypothesis holds for entity-oriented search (cf. Section 2.1.1 [73]), meaning that clusters around relevant nodes probably contain additional relevant nodes. Kloster and Gleich [165] also explored the heat kernel as a community detection algorithm, comparing it with the push algorithm by Andersen et al. [168] for approximating PageRank. They showed that the heat kernel coefficients decay quicker than the analogous damping factor from PageRank, ensuring shorter random walks. They solved the exponential of the Markov matrix, in the heat kernel equation, via a Taylor polynomial approximation, defining the *hk-relax* algorithm as a linear system. They were able to obtain consistently better results for *hk-relax*, when compared to the push algorithm, according to the F-measure and the conductance, over multiple graphs with a varying number of nodes and edges. More recently, Yang et al. [169] have also proposed the *TEA* and *TEA+* algorithms for a more efficient computation of the heat kernel PageRank. They proposed an approximation based on Monte Carlo random walks and a secondary algorithm to help reduce the number of required random walks. They were able to outperform the original algorithm by an order of magnitude for large graphs.

### 2.2.2 Text as a graph

For unstructured text, without hyperlinks, there are also models to represent documents as a graph of words. Blanco and Lioma [15] provide an in-depth exploration of graph-based models for text-based retrieval. They defined two graph-based representations of terms in a document, based on an undirected and a directed graph. The undirected graph linked co-occurring terms within a window of size N. Similarly, the directed graph also linked co-occurring terms within a window of size N, but established a direction based on grammatical constraints. This required POS tagging to be applied to terms and then, based on Jespersen's rank theory [37], POS tags were assigned a degree — 1st degree for nouns, 2nd degree for verbs and adjectives, 3rd degree for adverbs (and 4th degree for other tags). Under this model, higher rank words can only modify lower rank words. This relation was captured using a directed edge in the graph. Two raw metrics were then defined over each graph, using PageRank and the (in)degree to weight term nodes. This resulted in TextRank and TextLink over the co-occurrence graph (undirected), and PosRank and PosLink over the co-occurrence graph with grammatical constraints (directed). They then combined each raw term weighting metric with IDF for ranking documents according to the terms of a given query. This raw model was combined with several individual graph-based features, using the *satu* method by Craswell et al. [170], and retrieval effectiveness was assessed over TREC test collections (DISK4&5, WT2G and BLOGS06). Graph-based features added to the raw model included: average degree, average path length, clustering coefficient, and the sum of graph-based term weights (which worked as a type of document length normalization). The graph-based models were compared to the BM25 (the baseline), as well as TF-IDF, according to MAP, P@10 and bpref (Binary Preference). The best results for graph-based models were obtained for the BLOGS06 collection. Generically, the graph-based features improved the raw model and there was always a graph-based model that outperformed the baseline, although for some of them the difference was not statistically significant. They also measured the impact of the window size N, finding that $N = 10$ performed well for MAP and bpref, and they measured the impact on indexing time introduced by computing the graph-based features, finding that

TextRank only introduced an overhead of a few milliseconds (~50ms for 1,000 iterations).

Building on the previous work, Rousseau and Vazirgiannis [16] proposed a novel graph-based document representation, defying the term independence assumption of the bag-of-word approach. They defined an unweighted directed graph (the graph-of-word), where nodes represented terms, and edges linked each term to its following terms within a sliding window of size N, in order to capture context. Based on information retrieval heuristics [117, 118] and the graph-based term weighting approach by Blanco and Lioma [15], they also defined a retrieval model over the graph-of-word, based on the indegree of the nodes (TW-IDF). The goal of the weighting model was to measure the number of contexts a given term appeared in. They also introduced a pivoted document length normalization component, tunable with parameter b (analogous to BM25's b). The graph-of-word was generated per document, computing the TW metric and storing it within the inverted index, to be used as a replacement for TF. This meant that the document graphs could then be discarded without requiring persistence. They evaluated the TW-IDF ranking function with and without regularization over document length, as well as with and without parameter tuning for the pivoted document length normalization b parameter. They found that only a small contribution of document length normalization was required, thus settling on a constant value of $b = 0.003$. They also experimented with parameterizing the window size N, but since they didn't find an improvement for any of the tested values, they used a default value of $N = 4$. Finally, they did a comparison of TW-IDF with TF-IDF and BM25, as well as Piv+ and BM25+ (TF-IDF and BM25 with lower bound regularization [118]), showing that TW-IDF consistently outperformed the other weighting functions, particularly in realistic conditions, where parameter tuning is costly and is seldom an option.

In recent work, Dourado et al. [171] has come forth with a general graph-based model for text representation, able to support both the tasks of classification and retrieval. Their approach consisted of mapping text documents to a directed graph of words, capturing term order, and assigning node weights based on the normalized TF of the terms, and edge weights based on normalized TF of bigrams formed by the two words represented by the linked nodes. This was done for the whole collection, per document. For each document, subgraphs were then extracted, for example based on segments within a given path length, and then a vocabulary selection stage was carried based on a graph dissimilarity function and on graph clustering. Each cluster corresponded to a word (a centroid) in a codebook, representing the vocabulary that will be used to represent the documents. The subgraphs in each document graph were appointed to a centroid, either by hard assignment (the closest centroid), or soft assignment (based on a kernel function). The output of the assignment function was a matrix, where each vector represented the assignment weight to each centroid. A final pooling function then collapsed this matrix into a vector that represented the document graph, reaching the goal of graph embedding. From this point on, the vector could be used both for retrieval or classification, which the authors evaluated using multiple test collections. For the task of classification, their bag of textual graphs approach (as they called it), was able to outperform the remaining document representations for four of the five test collections, according to macro F1, which ranged from 0.676 to 0.997. For the task of retrieval, they experimented with the bag of textual graphs using three distances: Euclidean, Jaccard index, and cosine. They were able to outperform all baseline approaches, according to NDCG@10, when using the Jaccard and cosine distances, and most of them when using the Euclidean distance. The best results were obtained for the cosine distance. This work shows the reliance of graphs as a data structure to support general models. In this doctoral thesis, we explore a similar topic, but the application is to combined data, jointly representing corpora and knowledge bases, with the goal of supporting multiple entity-oriented search tasks.

### 2.2.3 Knowledge graphs

Instead of issuing direct queries over a graph, either by ranking its nodes (Section 2.2.1) or by matching subgraphs (Section 2.2.6), graph-based models can simply be used for the representation of knowledge in a retrieval process. We also consider such approaches to be graph-based, as long as there is an obvious and direct dependence on the entities and relations in a knowledge graph.

#### 2.2.3.1 *Augmenting entities with documents and vice-versa*

Fernández et al. [172] showed that ontology-based semantic search can be used for augmenting and improving keyword-based search. They proposed a system architecture for question answering based on natural language queries over the semantic web, using ranked documents to complement an answer given by ranked triples. The system relied on an ontology index, a concept-based index and a document index. The ontology index mapped terms to entities and was used both to build the concept-based index (document annotation) and for query processing (query annotation and triple matching). In particular, the PowerAqua system [173] was used for mapping keywords in a natural language query into triples from the indexed ontologies — they relied on WordNet to improve the matching between query terms and entities. The document index mapped terms to documents and was used for document ranking based on the retrieved triples and the concept-based index. Evaluation was performed using the TREC WT10G collection and a selection of 20 topics and their relevance judgments from TREC9 and TREC 2001. They also relied on 40 ontologies, based on Wikipedia, that covered the domain of the selected topics. Each TREC topic was expanded with an appropriate question answering request and additional information on available ontologies. They experimented with a baseline using a text-based approach over Lucene, semantic query expansion based on PowerAqua, and their complete semantic retrieval approach. When compared to the baseline, they obtained an improved effectiveness for 65% of the evaluated queries, according to average precision and P@10, when using their semantic retrieval approach, and 75% when considering only P@10 and either of the semantic approaches.

Byrne [174] dedicated her thesis to exploring the unified representation of hybrid datasets, combining structured and unstructured data, particularly in the domain of digital archives for cultural heritage. She relied on RDF triples, with a subject, predicate and object, to generate a graph that would integrate structured data from relational databases, unstructured data from entities and relations extracted from free text, and even domain thesauri useful for query expansion. For relational databases, each row in a table was instanced as a blank node of a class with the table name. For domain thesauri, the SKOS ontology was used to represent concepts and their relations of synonymy or hyponymy. For free text, 11 entity classes were considered, along with 7 predicates, one of which had a higher arity, containing 6 subpredicates that were used to establish binary relations. A classifier was trained for named entity recognition, and another one for relation extraction. Finally, equivalent queries were prepared to run over the RDF store as SPARQL, running either within Jena or AllegroGraph, and over the relational database as SQL, running within Oracle or MySQL. Byrne found that queries over RDF were considerably less efficient than queries over relational databases. She also found a lack of aggregation functions like *count* or *average* to query RDF, as well as the lack of graph theory functions to identify node degree or shortest paths. In this doctoral work, we also faced efficiency issues, which led us to implement approximated approaches. Given the importance of computing weights in information retrieval, we relied on graph and hypergraph theory to support ranking tasks in entity-oriented search, which is a feature that SPARQL does not support. Like Byrne, we are also interested in proposing a joint representation model for corpora and knowledge bases for maximizing the opportunities for a relevant answer to be found within all available data. Unlike Byrne,

we took the approach of openly exploring graph-based strategies, rather than restricting our work to RDF, exploring novel ways to represent text, entities, and their relations, in a way that supports retrieval by design.

Balog et al. [175] presented the SaHaRa entity-oriented search system for searching over news collections. For a given keyword query, SaHaRa used language models to retrieve both documents and entities, displaying them in a two-column interface. A document-centric view and an entity-centric view were also provided. The document-centric view was used to display a news article along with links to related articles and associated entities. The entity-centric view was used to display the entity, showing for example its Wikipedia summary, along with links to related news and Wikipedia articles, as well as associated entities, either based on the language model or the DBpedia relations. SaHaRa illustrates the benefits of augmenting documents with entities, as well as entities with documents, also showing that language models can be used for documents as well as entities.

#### 2.2.3.2 *Text-based retrieval of entities*

Blanco et al. [176] tackled the problem of effectiveness and efficiency in ad hoc entity retrieval over RDF data. Their ranking approach was based on BM25F, experimenting with three representation models: (i) an horizontal index, where fields *token*, *property* and *subject* respectively stored terms, RDF property names, and terms from the subject URI; (ii) a vertical index, where each field represented a separate RDF property name (e.g., *foaf:name*) containing terms from the respective literals; and (iii) a reduced version of the vertical index where fields represented *important*, *neutral* and *unimportant* values depending on the classification of the corresponding RDF properties. Evaluation was carried over the Billion Triple Challenge 2009 dataset [177]. For measuring effectiveness, they used the 92 entity-oriented topics and relevance judgments from the Semantic Search Challenge of 2010, obtained from Microsoft Live Search query logs. They compared BM25 from MG4J[1] with the three proposed indexes, finding the horizontal index to be the least efficient for *AND* and *OR* operators. Both the vertical and the reduced-vertical indexes were able to obtain a lower but comparable performance to BM25 for the *AND* operator, but not for the *OR* operator. Efficiency-wise, the best RDF index was the reduced-vertical. Regarding effectiveness, they compared BM25F with the BM25 baseline, as well as the best performing submission for SemSearch 2010. They found that, while the BM25 baseline was worse than the SemSearch 2010 baseline, their BM25F approach was able to improve MAP in 42% and NDCG in 52%. BM25F's $b$, field weight and document weight parameters were optimized using linear search and the promising directions algorithm [178], increasing MAP in over 35% just for tuning the parameter $b$ for each field. Increasing the weight of documents from *important* domains (e.g., dbpedia.org) was also significant.

Neumayer et al. [179] presented an overview on entity representation approaches for the text-based retrieval of entities. They covered the unstructured entity model, where all textual evidence was aggregated as a field in a virtual document, as well as the structured entity model, where textual evidence was aggregated in multiple fields, one per predicate type, in a virtual document. In particular, the aggregation into four predicate types was suggested: *Name*, *Attributes*, *OutRelations* and *InRelations*. Language models could then be applied to either representation and used as a ranking function, either over a single field or over the four individual fields. The presented models did not, however, preserve or take advantage of the information provided by individual predicates. Accordingly, the authors proposed the hierarchical entity model, where an entity was represented by the predicate types, as well the corresponding predicates. Additionally, each predicate type was represented both by its predicates and the text evidence for the type, and each predicate was represented by the text evidence for the predicate. Keyword based relevance for the

1 http://mg4j.di.unimi.it/

hierarchical entity model $\theta_e$ was then computed as illustrated in Equation 2.3, for an entity $e$, a predicate type $p_t$, a predicate $p$, and a term $t$.

$$\begin{aligned} P(t|\theta_e) &= \sum_{p_t} P(t|p_t, e)P(p_t|e) \\ &= \sum_{p_t} \left( \sum_{p \in p_t} P(t|p, p_t)P(p|p_t, e) \right) P(p_t|e) \end{aligned} \tag{2.3}$$

They also proposed four approaches for predicate generation $P(p|p_t,e)$ — *Uniform* (inverse frequency of predicates of the given type), *Length* (number of terms per predicate, normalized for the length of its predicate type), *Average length* (average number of terms per predicate, normalized for the average number of terms of its predicate type) and *Popularity* (fraction of triples with a given predicate, normalized for the number of triples containing any predicate of the same type). They found that the hierarchical entity model was able to outperform the unstructured entity model, but not the structured entity model. Perhaps more interestingly is that fact that it was able to fully capture the original semantic relations, without incurring in a significant loss of performance.

#### 2.2.3.3 *Building knowledge graphs*

Knowledge graphs have been powering mainstream web search engines since the 2010s. Bast et al. [92] highlighted DBpedia[1], YAGO[2] and Freebase[3], as some of the most relevant semantic web [2] resources.

**DBPEDIA** DBpedia is perhaps one of the most used public resources in the semantic web, frequently exploited to support search. Auer et al. [50] described the extraction process that transforms Wikipedia collaborative articles, as curated by the community, into a multi-domain knowledge graph, available through a SPARQL endpoint and interlinked with other open datasets. Examples of entity-oriented search that rely on DBpedia include for instance entity list completion by Bron et al. [128], or entity type ranking by Tonon et al. [180], as well as several examples across ad hoc document retrieval [181, 182] and ad hoc entity retrieval [179, 183, 184].

**YAGO** Standing for Yet Another Great Ontology, YAGO [95] is a well-known semantic web resource developed by the Max-Planck Institute for Informatics. Like DBpedia, it relies on information extraction over Wikipedia to obtain structured statements, but, through a set of rules and heuristics, it also integrates information from the WordNet lexical database, and from the GeoNames geographical names database. Like DBpedia, YAGO is frequently used to support several entity-oriented tasks, such as the annotation of the INEX 2009 Wikipedia collection [112], which was central in evaluating the graph-based models proposed in this doctoral work.

**GOOGLE KNOWLEDGE GRAPH** Google announced Google Knowledge Graph[4] in May 2012. Formerly, it had been partly powered by Freebase but, in July 2010, Google acquired Metaweb, the creators of Freebase, shutting down Freebase in August 2016. Data dumps were then made available by Google to the public, under the CC-BY license, and migrated to Wikidata[5], as a community-maintained alternative to Freebase. Knowledge bases can be used to semantically enhance text,

1 http://dbpedia.org/ontology/
2 http://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/
3 https://developers.google.com/freebase/
4 https://www.blog.google/products/search/introducing-knowledge-graph-things-not/
5 https://www.wikidata.org/

supporting tasks like named entity recognition or entity linking. They are central in entity-oriented search.

**MICROSOFT SATORI** In March 2013, Microsoft Satori[1] was announced. Then, in 2018, the Satori Group presented a tutorial on the 24th ACM SIGKDD Conference on Knowledge Discovery and Data Mining [185], where they described how the knowledge graph was built. Steps included data ingestion, match & merge, knowledge refinement and publishing & serve. They also proposed that the quality of a knowledge graph should by measured by correctness, coverage, freshness and usage.

**MICROSOFT ACADEMIC GRAPH** Sinha et al. [186] described the latest release of the entity graph used in the Microsoft Academic Service (MAS), after its integration with the Bing infrastructure. This led to a growth from under 10 million indexed papers to over 80 million indexed papers. Their heterogeneous graph was comprised of six types of entities: *#field_of_study*, *#author*, *#institution*, *#paper*, *#venue* and *#event*. Papers and authors were mainly discovered using feeds from publishers (e.g., ACM, IEEE), while venues, events and institutions were discovered from semi-structured websites that served as conference hubs and that were indexed by Bing. Practical applications of the described entity graph included: (i) serving constrained academic queries (e.g., `[ fields of study about Artificial Intelligence ]`), (ii) suggesting queries with the same prefix (e.g., `[ machine learning ]` and `[ machine learning algorithms ]`), and (iii) academic entity recommendation — for example, people related to a field of study (e.g., *Andrew Ng* would appear as an entity related to *Machine Learning*), or entities central to related queries (e.g., *Log-normal distribution* would appear as an entity searched along *Machine Learning*).

### 2.2.4 Entity graph from text

We have seen that both unstructured text and structured knowledge can be modeled as a graph. Beyond these individual representations, there are also approaches that focus on building entity graphs from text, establishing a direct relation between text and knowledge. This also helps to distinguish between knowledge that is internal and knowledge that is external to the collection. Bordino et al. [187] explored the topic of serendipity in entity search, evaluating results based on surprise and relevance, as well as based on interestingness. They created an entity network from Wikipedia and Yahoo! Answers based on the similarity of entities profiles built from the textual content citing an entity. In order to improve performance, they only compared pairs of entities that co-occurred in at least one document, based on the document similarity self-join algorithm by Baraglia et al. [188]. They then created an edge between two entities when their similarity was above a given threshold. For evaluation, they collected the most searched queries in 2010 and 2011 from Google Trends[2], identifying the entity associated with each query. The queries covered topics about people, places, websites, events, gadgets, sports, and health. They then used a crowdsourcing platform to obtain relevance judgments and indicators of interestingness, and they quantified surprise based on whether results appeared on commercial search engines, according to different criteria. Finally, serendipity was measured based on the normalized aggregated relevance of surprising results. They found that 51% of the nodes in the Wikipedia network overlapped with the nodes in the Yahoo! Answers network and also that both networks were nearly 95% connected, through the presence of common concepts that bridged the gaps. Their ranking method was based on the stationary (time-independent) distribution of lazy random walks in the graph, with a $\lambda = 0.9$ probability to stay in the input entity

1 https://blogs.bing.com/search/2013/03/21/understand-your-world-with-bing/
2 Google Trends is identified in the paper as Google Zeitgeist, which was a previous designation.

node, which at $d = 0.85$ worsened the results[1]. They also introduced three main constraints based on: (i) quality (measured through readability); (ii) sentiment (based on SentiStrength[2] as applied to the associated textual documents); and (iii) topic categories (using a proprietary classifier to identify 18 main categories). They then measured the fraction of unexpected (surprising) and relevant recommendations, over different runs, for unconstrained search, as well as considering topic, high sentiment, low sentiment, high readability, and low readability constraints. Overall, they obtained the best results for topic constrained search and for high readability constrained search. They showed that Wikipedia and Yahoo! Answers were good datasets for promoting serendipitous search, as they returned relevant results that were dissimilar to those found through other web search engines. Recommendation tasks as the one presented in this work are analogous to the entity-oriented search task of related entity finding, although ignoring the target entity type and experimenting with additional constraints.

Ni et al. [189] proposed a concept graph[3] representation of a document and the measurement of semantic similarity based on that graph. They used TAGME[4] to annotate documents with mentions linked to Wikipedia concepts. Then they built a graph using concepts as nodes and three types of concept relations as edges — *:context* (connecting concepts sharing incoming links from common Wikipedia articles), *:category* (connecting concepts belonging to similar categories in the Wikipedia taxonomy), and *:structure* (based on the graph induced by the links within the Wikipedia infobox of each concept and the shortest path between concepts). The *:context* and *:category* edges are similar to the bibliographic coupling and co-citation approaches described in Section 2.2.1, based on the work by Van and Beigbeder [162] to capture semantic relatedness. Each edge was weighted by a similarity metric proposed for each specific concept relation type. The authors also assigned weights to the nodes based on the closeness centrality for each node, using a custom distance metric defined by the inverse of a linear combination of the weights of the three possible types of edges. An additional weight was associated with each node, based on the TF-IDF similarity between the concept's Wikipedia article and the represented document. Then they defined a pairwise concept similarity, called *Concept2VecSim*, where *Concept2Vector* was inspired by word2vec, and a document similarity called *ConceptGraphSim* based on the best pairwise similarities of each concept of either document, relative to the concepts in the other document, as well as the weight of the concepts in the graph. They compared their methodology, optionally combined with ESA [139], with several other state-of-the-art methodologies, both through individual and combined applications. They concluded that their approach outperformed the majority of the methodologies, with the exception of *WikiWalk + ESA* [191] when compared with *ConceptGraphSim* alone, and *ConceptsLearned* [192] when compared with *ConceptGraphSim + ESA*.

### 2.2.5 Entity graph as a tensor

Zhiltsov and Agichtein [75] captured the latent semantics of entity-relations based on tensor factorization. They defined a tensor that described entity relations based on different predicates, represented as multiple adjacency matrices, one per predicate over the third dimension of the tensor. Tensor factorization was applied to the tensor, using the RESCAL algorithm [193], in order to obtain a matrix of latent entity embeddings and a tensor of latent factors. A listwise learning to rank approach, based on gradient boosted regression trees, was then used to optimize a ranking function according to NDCG. They considered term based features, as well

1 Please notice that we normalized the notation to be consistent over the document. Here, $\lambda = \beta$ and $d = 1 - \alpha$, when compared to the original paper.
2 http://sentistrength.wlv.ac.uk/.
3 Not to be confused with conceptuals graphs [190].
4 https://tagme.d4science.org/tagme/.

as structural features. Term based features relied on a multi-field document representation of the entity, enabling the retrieval of entities based on keyword queries. In particular, they did this based on a mixture of language models, as well as a bigram relevance score per field. Structural features were based on the entity embeddings from the tensor. In particular, they computed the cosine similarity, the Euclidean distance and the heat kernel between the embedding of a given entity and the embeddings of each of the entities in the top-3 using a baseline ranking. Their evaluation was based on 142 queries from the SemSearch Challenge from 2010 and 2011 and the Billion Triple Challenge 2009 dataset. They consistently obtained an increased performance of nearly 5%, for NDCG, MAP and P@10, when considering structural features.

### 2.2.6 Graph matching

One of the approaches to graph-based retrieval is the definition of graph queries (e.g., translated from keywords or natural language), that can be issued over a text graph or a knowledge graph. In the context of graph data management, Fletcher et al. [194, §1.4.1] classified graph queries into four categories: adjacency queries, pattern matching queries, reachability queries, and analytical queries. Adjacency queries consider nodes linked by an edge, as well as edges that share a common node, and they can even consider a k-neighborhood (i.e., linked nodes/edges at a distance k). Pattern matching queries consist of finding values for variables in a triple or sequence of triples (e.g., $\langle ?x, :friend, ?y \rangle$ should return pairs of friends). Reachability queries determine which nodes can be reached based on the given traversal restrictions (e.g., $\langle John, :friend^{+}, ?x \rangle$ will return friends of *John*, as well as friends-of-friends of *John*, and so on). Finally, analytical queries include queries that are based on aggregated computations over a graph, including average path length, connected components, community detection, clustering coefficient, or PageRank. Fletcher et al. [194, Ch.4] also covered the concepts of query relaxation and approximation as a way to manipulate the path structure in a graph query to enable a more flexible query processing. This is aligned with the need for better retrieval techniques over knowledge graphs that, unlike text-based retrieval, do not yet provide adequate approaches based on keyword or natural language queries. Entity-oriented search tackles this type of challenges, making search easier over unstructured and structured data.

Zhu et al. [58] and Zhong et al. [57] have proposed an approach to semantic search for entity ranking, through the matching of a query graph and a resource graph. The idea was developed based on conceptual graphs [190], having a direct translation to RDF graphs[1]. The conceptual graphs were built from natural language queries and documents via their prototype ALPHA [195]. They measured the similarity between two conceptual graphs based on the similarity between their nodes and edges. Node similarity was computed using WordNet[2], based on the distance to the closest common parent of two concepts. Concepts that subsumed each other were considered to have distance zero and thus similarity one. Edge similarity was computed as a binary value that was one, only when the edge from the query graph subsumed the edge from the resource graph. For the computation of graph similarity, they avoided the maximum subgraph matching problem, which is NP-complete, by defining entry nodes that the user should identify in their queries. As shown in Equation 2.4, graph similarity (*SoG*) was computed based on the node similarity of entry concepts ($c_Q$ and $c_R$) and on the edge similarity of the relations directly associated with the entry concepts ($r^j_Q$ and $r^j_R$) — recursion was then carried over subgraphs under each individual relation, using the node linked by that relation as the new entry node ($c_Q^{r^j_Q}$ and $c_R^{r^j_R}$). Optionally, user weights ($w(c_Q, \cdot)$)

1 https://www.w3.org/DesignIssues/CG.html
2 https://wordnet.princeton.edu/

could also be assigned to relations between entry nodes and their children, otherwise using a default value.

$$\begin{aligned} \mathrm{SoG}\left(c_Q, c_R\right) = {} & w\left(c_q, c\right)\ \mathrm{sim}_c\left(c_Q, c_R\right) + \\ & + \max_{\substack{\text{for every} \\ \text{combination}}} \left\{ \sum_j w\left(c_Q, j\right)\ \mathrm{sim}_r\left(r_Q^j, r_R^j\right)\ \mathrm{SoG}\left(c_Q^{r_Q^j}, c_R^{r_R^j}\right) \right\} \end{aligned} \tag{2.4}$$

Minkov and Cohen [196] were concerned with personal information management and the application of graph walks to derive entity similarity. They were able to use queries to generalize multiple tasks over an entity-relation graph (e.g., modeling e-mail as a graph of *people* who send and receive *messages* that contain *terms*). A keyword query was first processed in order to identify corresponding nodes in the graph, along with the target type of the output nodes. A response to the query consisted of a ranked list of entities of the given target type. Different tasks were defined based on the relations in the graph and could be specified along with the query. User feedback was also considered for learning task-specific similarities. This approach fits the tasks of related entity finding or entity list completion in entity-oriented search. Graph walks were based on personalized PageRank and, in particular, the alternative notation already presented in the context of the heat kernels used by Chung [164] and Kloster and Gleich [165]. They considered introducing walk bias based on the learned edge weights on a per-task basis. They also considered a reranking approach based on global features of the graph, such as the reachability from seed nodes (i.e., the count of source nodes that link to target nodes). They evaluated several personal information tasks, based on MAP and P@1, modeling them as queries over the graph. This included name disambiguation for people, grouping messages from the same thread, and finding e-mail aliases (based on people or messages in the graph). For the task of name disambiguation, they experimented with the Jaro similarity score [197] as a baseline, comparing it with graph walks using uniform weights, learned weights and reranking. Overall, the best results were obtained for the reranked version of the graph walks, with the exception of one dataset where the learned weights version performed better. For the threading task, they used the TF-IDF as the baseline, overall obtaining the best results for the reranked version of the graph walks. Finally, for the alias finding task, they also relied on the Jaro similarity score for the baseline, obtaining the best results for the graph walk.

Zhong et al. [198] have worked on keyword-based search over knowledge graphs. They combined content-based and structure-based features to score answer trees. In particular, weights were manually assigned to nodes, depending on the context, and then PageRank was used to balance and normalize the weights. Initial weights were also assigned to edges, computing their final weight based on the weights of the source and target nodes, as well as the initial weight. Answer trees were then extracted from the graph based on whether they contained the query keywords, and scored based on the distances between the root node and each query keyword. The distances were multiplied by a penalizing factor, that increased with the number of previous trees with the same root node, and then they were summed. The obtained score, where lower was better, was used to rerank the trees according to compactness — this combined content-based features from node popularity with structure-based features from graph relations. Evaluation was done over a DBLP[1] graph with 840 thousand vertices, 1.3 million edges, 95 thousand terms, and edge weights between 0.31 and 0.99. Their approach to evaluation was based on a manually built collection of keyword queries and a manual assessment of the rankings, with a focus on the diversity of the returned results.

Zhu et al. [199] proposed a natural language interface to a graph-based bibliographic information retrieval system. Through named entity recognition and de-

1 https://dblp.org

pendency parsing, they were able to generate a graph query that was capable of correctly interpreting 39 out of 40 natural language queries of varied complexities. The approach relied on a graph database to store the bibliographic data. A natural language query was then processed using named entity recognition to obtain the nodes for the graph query to be issued over the graph database. Dependency parsing was applied to extract relations between tokens (including entities), which were then adapted to the database schema, for instance adding missing nodes (e.g., the dependency $\langle papers, happy\ university \rangle$ might be translated into $\langle ?author, paper \rangle$, linked by a *:writes* relation, and $\langle ?author, happy\ university \rangle$, linked by a *:is_affiliated_with* relation). This abstract graph query could then be instantiated into a graph query language available for the graph database, where *?author* is a node of type *#author*. Despite the identified domain-dependent limitations of the model, this contributed to the application of graphs as a tool for natural language understanding and question answering.

Zhang et al. [200] explored graph-based document retrieval, by converting both documents and queries to graphs consisting of words and POS tags as nodes, and syntactic dependencies as edges. They segmented the documents into document semantic units (DSU), representing the atomic unit of parsing (e.g., a sentence, or a phrase within a sentence). They extracted graphs from each DSU, for each document. Node weights were computed from TF-IDF. They repeated this process for the query, considering it a single DSU. They then computed the maximum common subgraph between query and document graphs, taking into account that two nodes are the same if they share the same pair of word and POS tag. Node weights were combined based on the square root of the product, and edge weights were assigned based on whether the edge was present in both original graphs (1 if true and 0.5 if false). They calculated the similarity of a query graph and DSU graph based on a linear combination of the normalized sums of node and edge weights. The score for a document was computed based on the average of the similarities for all graphs representing document DSUs, or alternatively in a variant that assigned a higher weight to the title DSU. They prepared two datasets, one for Chinese and another one for English, by randomly selecting topics from the Sogou or Wikipedia top visited pages, and issuing those queries over Baidu or Google, in order to obtain result documents. Those documents were then graded for relevance by five human judges to form a test collection. Evaluation was done using DCG (Discounted Cumulative Gain) and the best results were obtained with the title-biased score function. The graph-based approach outperformed the vector space model for both the Chinese and English test collections, and it even outperformed the Google algorithm.

### 2.2.7 Hypergraph-based models

Hypergraphs [40] are a generalization of graphs, where edges (or hyperedges) can connect an arbitrary number of nodes — undirected hyperedges are represented by a set of nodes, while directed hyperedges are represented by a tuple of two sets of nodes. When all hyperedges in a hypergraph contain the same number $k$ of nodes, the hypergraph is said to be $k$-uniform. In that case, it can be represented as a tensor of $k$ dimensions, each of size $|V|$. In Section 2.2.5, we had covered tensor factorization over a tensor of entity relations for different predicates. Exploring analogous methods based on hypergraphs might also wield interesting results. A non-uniform hypergraph is called general. This is a family of hypergraphs that is rather hard to represent using tensors. CERN (Conseil Européen pour la Recherche Nucléaire) and the University of Geneva have recently been tackling this problem, focusing on undirected general hypergraphs [115], as well as hyperedges based on multisets instead of sets [201]. A hypergraph can also be called mixed [202, §4], when it contains both directed and undirected hyperedges, sometimes referred to as hyperarcs and hyperedges, respectively. In this thesis, we work with general mixed hypergraphs, using the designation of directed and undirected hyperedges.

Since, to our knowledge, there is no adjacency tensor representation for this family of hypergraphs, our implementations rely on existing data structures and software libraries, rather than linear algebra approaches that depend on an adjacency tensor.

Hypergraphs have also been explored outside of mathematics. In information science, topic maps [203, 204] have been used to jointly represent multiple indexes. Conceptually, topic maps are hypergraphs where nodes are topics or occurrences, and hyperedges are binary connections between topics and occurrences, or n-ary connections between topics. Garshol [203] has showed that topic maps can be used as a common reference model to represent metadata and subject-based classification, including controlled vocabularies, taxonomies, thesauri, faceted classification and ontologies. This means that, not only can topic maps be used to merge indexes, but also to extend the indexes with external knowledge, in order to improve search. While information retrieval was identified by Garshol as one of the main applications of topic maps, to this date not many actual applications can be found outside of information science. Yi [204] compared thesaurus-based information retrieval with topic-map-based information retrieval, by measuring the recall and search time of 40 participants over the two systems. He distinguished between queries based on a single concept (fact-based) and queries based on two or more concepts (relationship-based). They found that the topic-map-based system outperformed the thesaurus-based system, both regarding recall and search time, for relationship-based queries.

While hypergraphs have been previously used in information retrieval, they still don't play a major role in well-known tasks, despite their potential, as identified for instance in topic models. Perhaps the most notable work on hypergraphs for information retrieval is the query hypergraph proposed by Bendersky and Croft [14]. In the query hypergraph, nodes represent concepts from the query, and edges represent the dependencies between subsets of those nodes and a document. The query hypergraph is therefore able to represent higher-order term dependencies, capturing the "dependencies between term dependencies". Two types of hyperedges were defined: *local*, between individual concepts and the document; and *global*, between the entire set of concepts and the document. In order to obtain a score for a document and query, they relied on a factor graph representation of the hypergraph — a bipartite graph, where each hyperedge was represented by a factor node. The ranking function was then computed based on the local and global factors, that worked as document-dependent hyperedge weights. The approach is similar to other log-linear retrieval models, such as the Markov network model or the linear discriminant model, however higher-order term dependencies are easier to incorporate into the model. Their methodical approach can be regarded as a fundamental step in supporting hypergraph-based work in information retrieval.

In entity-oriented search, we frequently deal with combined data or, at the very least, we separately work with corpora and knowledge bases. Accordingly, finding a joint representation for this kind of unstructured and structured data represents added value for designing general solutions that solve information needs. Menezes and Roth [205] have recently introduced semantic hypergraphs, proposing an approach to represent knowledge extracted from corpora based on recursive ordered hypergraphs. On one side, such extension of hypergraphs means that nodes, representing terms, can now have an order in the hyperedge they belong to, enabling for instance the representation of an entity mention to be stored using the correct sequence of words. On the other side, recursivity means that higher-order dependencies are explicitly stored rather than being exclusively verifiable, enabling a hyperedge to be defined over nodes but also over hyperedges. This work is also available as a Python library called Graphbrain[1], which can be used to manipulate semantic hypergraphs for natural language understanding, and knowledge inference and exploration.

1 https://graphbrain.net

Recently, Dietz [103] has also proposed ENT Rank, a hypergraph-based approach for entity ranking, where text was used to inform and improve entity retrieval. The hypergraph was then converted into an entity co-occurrence multigraph and several features were considered to train a learning-to-rank-entities model: neighbor features, relation-typed neighbor features, and context-relevance features. The model was inspired by random walks with restart, where training consisted on optimizing two weight vectors, $\vec{\psi}$ and $\vec{\theta}$, as part of an equation similar to PageRank's, which acts as the ranking function for a given entity. In this equation, the features for the scored entity corresponded to the teleport or restart term, while the features from the neighbors and context corresponded to the the navigation term. The author's evaluation was based on the entity retrieval task from the TREC Complex Answer Retrieval track. It relied on the CAR dataset, with 5.41 million Wikipedia pages, along with a large corpus of paragraphs with hyperlinks to Wikipedia pages. DBpedia-Entity v2 was also used, with relevance judgments from SemSearch ES, INEX-LD, List Search and QALD-2. ENT Rank was able to achieve first or second best ranking model for all experiments, showing, in multiple cases, the best performance for unsupervised ranked aggregation.

Hypergraphs have also been recently used for summarization [206], as an XML alternative for the semi-structured representation of text as a graph [76] or to model folksonomies, promoting the serendipitous discovery of new content [207]. Even in 1981, in the area of social network analysis, Seidman [208] had noticed the inability of anthropologists and sociologists to study social networks based only on dyadic relationships, proposing hypergraphs as a way of better modeling non-dyadic relationships, such as group membership. Moreover, hypergraphs have already been particularly useful in music recommendation [83, 209–211] through unified approaches for modeling heterogeneous data or through the use of random walks. Given their ability to represent polyadic relations that group multiple nodes, hypergraphs have also been used as a way to compress semantic graphs [212].

Assuming that we would be able to effectively represent text and entities using a hypergraph, then we might be able to take advantage of both set theory, using metrics like the Jaccard index to measure similarities, or random walks in hypergraphs [55], where we might rely on hyperedge weights, but also on node weights to control the traversal. While hypergraphs are a flexible data structure, they still present some limitations, when applied to more complex representation needs. For instance, weights associated with nodes and hyperedges might not be enough to represent all types of bias — e.g., we can define node weights, but not node weights per hyperedge. There are, however, extensions of hypergraphs, like fuzzy hypergraphs [213], intuitionistic fuzzy hypergraphs [214] or hypergraphs with edge-dependent vertex weights [215], that provide increased flexibility in establishing bias. According to Canfora and Cerulo [216, Fig.1], this would in fact mean that such a model would simultaneously provide reasoning with logic (graph theory) and uncertainty (fuzzy set theory). Besides hypergraphs, there are also other higher-order data structures, like higraphs [217], hypernetworks [218, 219] or metagraphs [220], that might be worth exploring in information retrieval.

In prior sections of this chapter, we have already seen that graphs can be used to represent both unstructured text (e.g., graph-of-word [16]) and structured knowledge (e.g., DBpedia [50]). Hypergraphs can go even further, capturing for instance synonyms as undirected hyperedges. Moreover, approaches like hypergraph embeddings [221] can also be used to further reduce search complexity. The expressiveness and viability of hypergraphs make it a useful data structure to be explored in entity-oriented search. This doctoral work largely relies on hypergraphs to propose a general model for several retrieval tasks in entity-oriented search.

### 2.2.8 Random walk based models

Traversing a graph can be done through algorithms like breadth-first or depth-first search. For large graphs, however, the cost of using such strategies can be prohibitive. There are other less expensive traversal strategies, like random walks [222], that are still able to capture structural properties, but rely on a sampled view of the graph [223]. For example, while breadth-first search has time-complexity $O(|V| + |E|)$, a random walk has time-complexity $O(\ell)$, for a given length $\ell$, while also being easily parallelizable [224]. Accordingly, random walks frequently provide a more efficient way to estimate network properties. They can be used for measuring node importance, when applied globally (e.g., *PageRank* [113]), but also for community detection, when confined to local neighborhoods (e.g., *Walktrap* [225], push algorithm [168]), and they can even be used for entity linking, when applied to graphs of mentions and entities [226][227, §3.2.4].

PageRank is perhaps the most well-known and versatile graph-based metric that relies on random walks. It first surfaced in 1997, in a working paper by Larry Page and Sergey Brin [33], but it is usually cited using the 1998 article describing the Google search engine [151], or the 1999 technical report from Stanford InfoLab [113]. Since then, PageRank has been extended and reimagined by different researchers, who proposed their own improvements, as we have shown in Section 2.2.1. Experiments included measuring the importance of web pages based on a given topic [156], or considering a weighted approach based on network, semantic and visual features [157], or even introducing higher-order dependencies for modeling historical surfing information [114]. For further information, Appendix A provides an in-depth analysis of diverse PageRank variants and applications. There are also multiple available surveys about PageRank, namely from Chung [228] and from Gleich [155]. Chung [228] focused on approximated approaches for the computation of PageRank, also covering the applications and generalization of PageRank. Gleich [155] provided an in-depth survey with a good coverage on existing PageRank variants and applications, discussing the mathematics of PageRank and its generalizations. Our appendix section attempts to directly provide the calculations to obtain the Markov matrix necessary for computing each PageRank variant using power iteration. There are, however, multiple approaches for efficient PageRank computation, either based on speeding up power iteration [229, 230], parallel computing [231–233] or Monte Carlo methods [154, 234].

Due to its popularity, there are multiple applications of PageRank to entity-oriented search [62, §4.6.2]. In the remainder of this section, we present ReConRank, ObjectRank, HubRank and HopRank, with applications over RDF graphs or general labeled graphs. We present PopRank and DING, with applications over the semantic web, combining a web or dataset graph with an object or entity graph. Finally, we cover a semantics-aware personalized PageRank that explores PageRank for recommendation tasks, while considering RDF triples for improved performance.

**RECONRANK** Inspired by PageRank, Hogan et al. [66] proposed ResourceRank, ContextRank, and a combination of the two approaches called ReConRank. Using a similar strategy to the base set selection in HITS [149], a query dependent graph was built by matching RDF literals and returning their neighborhood graph. Equation 2.5 was then applied over multiple iterations $t$, until convergence, to rank an RDF resource $i \in V$, using information from the outdegree $deg^+(v)$, as well as the set of incoming neighbors $N^-(v)$, for a node $v$.

$$\begin{aligned} min_t &= \frac{1-d}{|V|} \sum_{j \in \{v | deg^+(v) \neq 0\}} rcr_{t-1}(j) + \frac{1}{|V|} \sum_{j \in \{v | deg^+(v) = 0\}} rcr_{t-1}(j) \\ rcr_t(i) &= \sum_{j \in N^-(i)} \left( \frac{d}{deg^+(j)} rcr_{t-1}(j) \right) + min_t \end{aligned} \tag{2.5}$$

The resource graph was induced by the nodes that appeared at least once as a subject in the retrieved RDF quads, while the context graph was induced by the fourth elements in the same set of RDF quads. In practice, each graph represented different projections of the original graph. ReConRank was then proposed as a metric computed over the combined graph of resources and their contexts, which represented an enriched connectivity over either individual graph. Equation 2.5 was used for the computation of either of the three metrics. We could say that these are representation-driven approaches, since the ranking function is unchanged, but applied to different graphs.

**OBJECTRANK** Balmin et al. [65] proposed an adaptation of PageRank for keyword search over a database modeled as a labeled graph. Like HITS, ObjectRank was computed over a base set to generate a topic-induced graph. Their approach consisted on precomputing the Global ObjectRank (same as PageRank) and the ObjectRanks for all term-based graphs, according to Equation 2.6, where $S(w)$ represents the base set for term $w$ and $s$ is the corresponding binary personalization vector.

$$OR^w = \frac{1-d}{|S(w)|}\, s + d\, \mathcal{M}\, OR^w \tag{2.6}$$

The keyword-based ObjectRank computed for a term could then be affected by the Global ObjectRank, $OR^G$, based on Equation 2.7, where the exponent $g$ controls its weight.

$$or^{w,G}(i) = or^w(i)\ (or^G(i))^g \tag{2.7}$$

For a query with multiple keywords, we can compute the product of individual ObjectRanks as the logical *AND* or, for pairs of keywords, the sum of ObjectRanks minus their product as the logical *OR*. It is also easy to derive the computation of ObjectRank for any combination of these boolean operators. A relevant difference between the computation of PageRank and ObjectRank, besides its query dependence according to the base set, is that the sum of outgoing weights, used to generate matrix $\mathcal{M}$, might be less than one. Weights are defined according to an authority transfer schema graph, where they are established for particular edge labels and between specific source and target node labels. Given that these weights might not add to one, the authors use the analogy of a random surfer that eventually disappears. For computational purposes, each weight is then divided by the weighted outdegree over edges with the same label, ensuring stochasticity and convergence (this normalization is similar to the one used in Equations A.9 or A.12).

**POPRANK** Nie et al. [235] have proposed a link analysis metric with applications to entity ranking, namely in academic search engines like Libra [236] — know, since 2011, as Microsoft Academic Search. PopRank acknowledged the importance of both the web graph (based on hyperlinks) and the object graph (based on heterogeneous relations between different types of objects). It combines web popularity, as well as transitions over the object graph, according to a popularity propagation factor. The popularity propagation factor $\gamma_{YX}$ was defined for links between two specific entity types $Y$ and $X$ (similar to the authority transfer schema graph in ObjectRank). The web popularity $WebPop_X$ was calculated based on the PageRank of the pages that contained the object, as well as based on the importance of web blocks (visual fragments of a web page). Equation 2.8 illustrates the calculation of PopRank $PopR_X$ for all nodes of type $X$.

$$PopR_X = (1-d)\ WebPop_X + d \sum_Y \gamma_{YX}\ M_{YX}\ PopR_Y \tag{2.8}$$

Notice that matrix $M_{YX}$ might not be square and therefore cannot be considered stochastic. However, it behaves quite similarly to a stochastic matrix, except it models the transitions from nodes of type $Y$ to nodes of type $X$. Nevertheless, when multiplied by $\mathrm{PopR}_Y$, the resulting vector will always be of size $|X|$. According to the equation, PopRank measures the popularity of a given object or entity by taking into consideration its popularity in the web, as well as the influence from different types of nodes, based on their specified propagation factor.

**HUBRANK** Chakrabarti [67] proposed an efficiency improvement over ObjectRank, where the personalization vector was only computed for a set of hub nodes selected based on query logs. They proposed a *TypedWordGraph*, where they introduced word-to-entity relations, thus enabling mixed word and entity queries. Each vector was approximated using precomputed fingerprints — i.e., the end nodes from random walks of various lengths, as sampled from a geometric distribution, and initiated from each node — as described by Fogaras et al. [237]. In order to compute HubRank, a subgraph limited by boundary nodes was first prepared. The boundary was established by a subset of hub nodes called blockers, and by loser nodes that were too far to significantly influence the personalized PageRank of the word nodes. Personalized PageRank was then estimated for the remaining active nodes and iteratively computed using dynamic programming, while fixing the value of boundary nodes. Fingerprinting and computation over a smaller graph provided improved efficiency, while the word-to-entity relations provided a more flexible model for entity-oriented search.

**DING (DATASET RANKING)** Delbru et al. [72] proposed a hierarchical link analysis approach based on the computation of a PageRank variant called DatasetRank, applied over a two-layer model of the semantic web. DatasetRank combines a local entity rank, indicative of the importance of an entity within the current dataset, with the the probability of jumping to another dataset, which is dependent on its size. Equation 2.9 illustrates the computation of DatasetRank for dataset $D_j \in \mathcal{D}$, based on its entities $E_{D_j}$ and incoming neighbors $D_j$. Each DatasetRank for neighbor datasets is then weighted based on a Link Frequency $\times$ Inverse Dataset Frequency (LF-IDF) computed over the set of links $L_{\sigma,i,j}$ with label $\sigma$, source node $i$ and target node $j$.

$$
\begin{aligned}
w_{\sigma,i,j} &= \mathrm{LF}(L_{\sigma,i,j}) \times \mathrm{IDF}(\sigma) = \frac{|L_{\sigma,i,j}|}{\sum_{L_{\tau,i,j}} L_{\tau,i,j}} \times \log \frac{|\mathcal{D}|}{1 + \mathrm{freq}(\sigma, \mathcal{D})} \\
\mathrm{dsr}_{t+1}(D_j) &= (1-d)\frac{|E_{D_j}|}{\sum_{\mathcal{D}} |E_D|} + d \sum_{L_{\sigma,i,j}} w_{\sigma,i,j}\ \mathrm{drs}_t(D_i)
\end{aligned}
\tag{2.9}
$$

**SEMANTICS-AWARE PERSONALIZED PAGERANK** Musto et al. [238] have experimented with personalized PageRank for recommendation over different user preference graphs, adding to the user-item relations with external knowledge from linked open data. Their contribution was focused on finding the best representation model for semantics-aware recommendation using personalized PageRank, rather than proposing changes to PageRank as a ranking function. They experimented with the bipartite user-item graph, as well as the tripartite user-item-resource graphs, based on all DBpedia triples, as well as on a subset of triples selected using PCA or information gain. They also experimented with different weighting schemes for each node type. They found slight benefits to the extension of user-item graphs with linked open data, particularly for graphs that were originally sparser.

**HOPRANK** Espín-Noboa et al. [239] proposed HopRank to model human navigation on semantic networks. Based on the analysis of user behavior in the BioPortal

**Table 2.1:** Chronological summary of entity-oriented PageRank variations.

| PageRank | Year | Highlights |
|---|---|---|
| ObjectRank [65] | 2004 | • Keyword search over databases modeled as labeled graphs.<br>• Each node contains properties used for keyword matching.<br>• Computed over a query-dependent graph like HITS.<br>• Precomputed for graphs based on individual keywords.<br>• Also precomputed for the whole graph.<br>• Edge weights defined in an authority transfer schema graph.<br>• Both directions are considered; might have different weights.<br>• Relevance score from global and keyword-based ObjectRanks.<br>• Boolean AND/OR operators defined for multi-keyword queries. |
| PopRank [235] | 2005 | • Used for link analysis in Libra (Microsof Academic Search).<br>• Object-level ranking.<br>• Based on the web graph and the object graph.<br>• Edge weights defined for predicates (single direction).<br>• Web popularity based on PageRank and web block importance.<br>• Transition matrices between nodes of different types. |
| ReConRank [66] | 2006 | • ResourceRank over a resource graph.<br>• Based on links between entities used as a subject.<br>• ContextRank over a context graph.<br>• Based on links between the context given by quads.<br>• Contexts linked when they mention a common entity.<br>• ReConRank over a unified graph. |
| HubRank [67] | 2007 | • Designed to improve the performance of ObjectRank.<br>• Computed for a set of hub nodes selected from query logs.<br>• Based on a *typed word graph*.<br>• Considers term-entity and entity-entity relations.<br>• Identifies boundary nodes (blockers and losers).<br>• Uses fingerprints based on random walks to estimate scores. |
| DING [72] | 2010 | • Dataset rankING combines a DatasetRank and a local entity rank.<br>• Described as hierarchical link analysis.<br>• Two levels: dataset graph and entity graph.<br>• Local entity rank as the fraction of entities per dataset.<br>• Combines with the weighted influence of incoming datasets. |
| Semantics-Aware Personalized PageRank [238] | 2017 | • Recommendation over preference gaphs.<br>• Bipartite user-item graph.<br>• Tripartite user-item-resource graph.<br>• Extended with DBpedia triples.<br>• Node-based weighting scheme. |
| HopRank [239] | 2019 | • Used for ranking ontology classes.<br>• Teleportation studied over $k$-hops.<br>• *HopPortation* probabilities based on clickstream transitions.<br>• Transition matrices for each $k$-hop.<br>• Ignores ontology edge direction.<br>• Defines one teleportation term.<br>• And $d'$ $k$-hop elements, for an ontology with diameter $d'$. |

website[1], a repository of biomedical ontologies, they found that, instead of teleporting to random ontology nodes, users showed a bias toward jumping to nodes at a particular distance $k$. They called this a $k$-hop, naming the probabilities of teleporting to $k$-hops as *HopPortation*. Given the diameter $d'$ of the ontology (ignoring direction), consider the *HopPortation* vector $\vec{d}$ of size $d'+1$, where $d_k \in \vec{d}$ represents the probability of a $k$-hop happening. The authors computed $d_k$ based on the clickstream transitions in the BioPortal website, using add-one smoothing to ensure each available $k$-hop was considered. Also consider $d'$ matrices $\mathcal{M}_k$ containing the transition probabilities for the corresponding $k$-hops, based on the undirected ontology links. HopRank is then calculated based on the Markov matrix $\widehat{\mathcal{M}}$ described in Equation 2.10.

$$\widehat{\mathcal{M}} = \frac{d_0}{|V|} + d_1\mathcal{M}_1 + d_2\mathcal{M}_2 + \ldots + d_k\mathcal{M}_k \tag{2.10}$$

Table 2.1 provides a chronological overview of PageRank adaptations and applications to entity-oriented search and related tasks. As we can see, each approach uses different nomenclature. Overall, however, the majority of the approaches are based

1 https://bioportal.bioontology.org/

on a combination of entity graphs (resource graph, labeled graph, object graph) and dataset graphs (context graph, web graph, dataset graph). There are also cases where the ontology graph (not unlike an entity graph) is directly used, or where a combined graph of users, items and resources (again, similar to an entity graph) is used. These last two cases represent work that, while not directly applied to entity-oriented search, can be easily adapted, given the usage of underlying graphs with semantic characteristics.

## 2.3 EVALUATION METHODS AND RESOURCES

Traditionally, information retrieval follows an empirical approach to research, relying on experimentation over test collections to assess the quality of retrieval models [240]. Evaluation forums, such as TREC, INEX (INitiative for the Evaluation of XML Retrieval), or CLEF (Conference and Labs of the Evaluation Forum) also bring the community together to prepare these datasets, offering multiple evaluation moments, under the same conditions, for registered researchers. Each event usually has a list of tracks, available to participants — tracks approach specific retrieval tasks, usually providing a dataset or system for evaluation. In this section, we begin by covering contributions that illustrate archetypal evaluation approaches, and we cover some of the most relevant test collections and evaluation forums for the assessment of information retrieval tasks, in particular focusing on entity-oriented search.

### 2.3.1 Evaluation approaches

In entity-oriented search, evaluation approaches are in line with the overall information retrieval research methodology, relying on test collections, where topics can either be used to build keyword or entity queries, and relevance judgments are specific to the tasks, where either documents or entities are graded. In this section, we illustrate two evaluation approaches for entity ranking tasks.

Komninos and Arampatzis [241] presented a web application for entity ranking that receives a query in natural language and identifies the most relevant entities associated with the query. For evaluation, they used the topics from the entity ranking tracks from INEX 2009 and TREC 2010. They tested the effectiveness of eleven ranking alternatives, discovering that the number of documents that cite an entity is more relevant than the number of times the entity is cited in the documents. They also found that in the top-$n$ retrieved documents, when considering a small $n$, document rank information has little influence over entity relevance. They verified that the best results were achieved when using the maximum entropy algorithm with a scoring function that combined the logarithmic entity frequency with the document frequency.

Blanco et al. [242] created a standardized evaluation setting for entity search, based on three test collections: (i) Billion Triples Challenge 2009 dataset; (ii) a subset of the Yahoo! Search Query Tiny Sample v1.0 dataset, where 50 queries containing an entity were handpicked; and (iii) crowdsourced relevance assessments from Amazon Mechanical Turk, where human evaluators were given a keyword query along with an entity ranking, displayed as a table representation of the RDF triples about the entity, judging it based on a three-point scale as *irrelevant*, *somewhat relevant*, or *relevant*. This evaluation setting was applied in the context of the 2011 SemSearch Challenge [243] — queries were targeted at structured data in RDF rather than text in unstructured web pages, like TREC 2010 Entity track [69].

Table 2.2: Datasets for entity-oriented search: corpora, knowledge bases, and combined data.

| Dataset | Description |
|---|---|
| | **Unstructured data:** corpora (e.g., plain text, hypertext) |
| AQUAINT 2002/2008 | Collectively consists of almost two million English news articles, from sources that include New York Times, the Associated Press, or the Xinhua News Agency newswires, spanning from 1996-2000 and 2004-2006. |
| BLOGS06 | Blog posts collected from 100k RSS and Atom feeds, spanning from 2005 to 2006. |
| ClueWeb 2009/2012 | Nearly 1 billion web pages, spanning from January to Feburary 2009. |
| CommonCrawl 2007 | Petabytes of data for a period of seven years. Contains web pages, along with extracted metadata and plain text. |
| KBA Stream Corpus 2014 | Consists of timestamped data for links (unshortened bitly URLs), social (blogs and forums), and news. It contains over 5.4 million links, 322.6 million social documents, and 134.6 million news. |
| TREC Washington Post Corpus (2017) | Contains 608,180 news articles and blog posts from January 2012 through August 2017 in JSON. Topics are provided for the Common Core track, covering the ad hoc document retrieval task, as well for the News track, covering the tasks of background linking, and entity ranking. |
| | **Structured data:** knowledge bases (e.g., triples) |
| BTC 2009/2010/2012 | Billion Triples Challenge dataset. The 2009 version contains over 1.14 billion statements, consisting of over 1.46 billion nodes, 866 billion resources, 352 billion blank nodes, and 246.7 billion literals. |
| | **Combined data:** corpora and knowledge bases (e.g., annotated text) |
| ClueWeb09 FACC, ClueWeb12 FACC | Freebase annotations of the ClueWeb Corpora, as done automatically by Google researchers. Annotations represent entities with a high level of confidence. On average, there are 15 entity mentions for ClueWeb09 and 13 entity mentions for ClueWeb12. |
| INEX 2009 Wikipedia collection | Over 50 GiB of uncompressed XML files representing Wikipedia articles annotated with over 5,800 entity classes. This dataset is accompanied by a set of topics from 2009 and 2010, along with the respective user assessments, for multiple entity-oriented search tasks. Given it is also publicly accessible, it is one of the best datasets available for evaluation, despite its age. |
| INEX Wikipedia LOD Collection | Dataset of combined data, with over 12.2 million XML articles, accompanied by a list of DBpedia URIs for included articles. |
| | **Other datasets** |
| Wikipedia, Wikidata, DBpedia, FAKBA1, Web Data Commons (WDC) 2012, WordSim353, SimLex-999, MEN, Rare words, Freebase Easy, Free917, WebQuestions, Mondial, IMDB, WDC / SemSearch Challenge 2010 queries, MusicBrainz. | |

### 2.3.2 Test collections

Datasets are a fundamental resource for the testing and development of entity-oriented search. Campinas et al. [244] provided a dataset called Sindice–2011, with over 230 million documents and 1.7 billion entities. While the dataset seems to have gained popularity within the community, official web sites[1] are down at the time of the writing of this document and have been down at least from the beginning of the ANT[2] project (June 2015).

Bast et al. [92] covered several datasets useful to the overall area of semantic search. Table 2.2 presents a list of relevant information retrieval test collections, covering corpora and knowledge bases, as well as combined data, which is of special

1 http://data.sindice.com/trec2011/ (broken link)
2 http://ant.fe.up.pt/about

**Table 2.3:** Events and respective tracks, tasks or challenges relevant to entity-oriented search.

| Event | | Track / Task / Challenge | |
|---|---|---|---|
| TREC | Text REtrieval Conference (1992-ongoing) | KBA | Knowledge Base Acceleration Track (2012-2014) |
| | | – | Entity Track (2009-2011) |
| | | QA | Question Answering Track (1999-2007) |
| | | Live QA | Live QA Track (2015-2017) |
| INEX[2] | INitiative for the Evaluation of XML retrieval (2002-2014) | – | Ad Hoc Track (2007-2010) |
| | | – | Entity Ranking Track (2007-2009) |
| | | – | Linked Data Track (2012-2013) |
| | | – | └Jeopardy! Task (2012-2013) |
| CLEF | Workshop of Cross-Language Evaluation Forum / Conference and Labs of the Evaluation Forum (2000-ongoing) | QA | Multilingual Question Answering Task (2003-2008) |
| | | WiQA | Wikipedia Question Answering Task (2006) |
| | | WSD QA | Word Sense Disambiguation for Question Answering (2008) |
| | | WePS | Searching Information about Entities in the Web (2010) |
| | | | └Web People Search Task (2010) |
| | | QA4MRE | Question Answering for Machine Reading Evaluation (2011-2013) |
| | | ER | Entity Recognition Challenge (2013) |
| | | QALD[1] | Question Answering over Linked Data (2011-2014) |
| | | INEX[2] | INitiative for the Evaluation of XML retrieval (2012-2014) |
| | | BioASQ | Biomedical Semantic Indexing and Question Answering (2015) |
| TAC | Text Analysis Conference (2008-ongoing) | KBP | Knowledge Base Population Track (2009-2018) |
| | | QA | Question Answering Track |
| SIGIR | Special Interest Group on Information Retrieval (1978-ongoing) | ERD | Entity Recognition and Disambiguation Challenge (2014) |
| QALD[1] | Question Answering over Linked Data (2011-2018) | – | Hybrid Question Answering (2014,2017-2018) |
| | | – | Multilingual Question Answering (2013) |
| | | – | Ontology Lexicalization (2013) |
| | | – | Multilingual QA over Linked Data (2014) |
| | | – | Biomedical QA over Interlinked Data (2014) |
| | | – | Multilingual QA over RDF Data (2016) |
| | | – | Hybrid QA over Both RDF and Free Text Data (2016) |
| | | – | Statistical QA over RDF Data Cubes (2016) |
| | | – | Multilingual QA over DBpedia (2016-2018) |
| | | – | English QA over Wikidata (2017) |
| | | – | Large-Scale QA over RDF (2017) |
| SemSearch | Semantic Search Workshop (2008-2011) | – | SemSearch Challenge (2010-2011) |

[1,2] Same event.

interest to entity-oriented search. In particular, we highlight the INEX 2009 Wikipedia collection, which is a combined data test collection that was used for several tasks in the Ad Hoc track and the Entity Ranking track from INEX. As we show further along, this dataset became a fundamental resource for experimenting with a universal ranking function over multiple retrieval tasks.

### 2.3.3 Evaluation forums

Bast et al. [92] also covered several evaluation forums that support the area of semantic search. Table 2.3 lists the most relevant evaluation forums, along with a selection of tracks and tasks of special interest to entity-oriented search. In particular, we highlight TREC Entity track, which ran from 2009 to 2011, and INEX Entity Ranking track, which ran from 2007 to 2009. In the following sections, we delve into further detail on two of the most relevant evaluation forums in the community, TREC and INEX, establishing a parallel between the entity tracks in the two events.

#### 2.3.3.1 *TREC – Text REtrieval Conference*

The Text REtrieval Conference (TREC) began in 1992 and it has ever since brought together the Information Retrieval (IR) community to participate in several research

tracks. Each research track represents a different open challenge in the area. Tracks can be discontinued when a problem has been solved or interest has faded, and new tracks are frequently created or transformed to better represent new or relevant challenges. TREC participants are expected to choose one or multiple tracks, each providing specific resources (e.g., document collections, relevance judgments, APIs, etc.), in order to develop a search engine that will be evaluated on a common framework. The event is organized as an evaluation forum rather than a typical conference. As the result of a TREC participation, a research paper must be submitted to publish in a NIST Special Publication dedicated to TREC, describing the approach taken and the obtained results. The focus is on using traditional IR metrics, such as Mean Average Precision (MAP) or Normalized Discounted Cumulative Gain (NDCG), to measure the quality of different models or different parameter values based on human relevance judgments. Most tracks also provide an overview of the occurrence, where the quality of the participants' runs is compared (e.g., Balog et al. [142], for the 2011 Entity track). This works as a state-of-the-art assessment of a particular information retrieval task. Next, we present an overview of the tracks that we have identified as relevant for entity-oriented search in Table 2.3.

**KNOWLEDGE BASE ACCELERATION** Knowledge Base Acceleration (KBA) ran consecutively from 2012 to 2014 and was succeeded by the Dynamic Domain track in 2015, which also ran in 2016 and in 2017. KBA[1] describes their mission as follows: "Given a rich dossier on a subject, filter a stream of documents to accelerate users filling in knowledge gaps.". This track tackled the challenge of increasing the speed at which a news article is cited in a knowledge base from the moment of its publication. In 2014, the median number of days a news article waited to be cited in Wikipedia was 356 days (nearly a year!)[2]. The KBA track dealt with this issue by focusing on improving entity-oriented filtering of large streams, in order to help human curators fill in the knowledge gaps more quickly. Tasks relied on the TREC KBA Stream Corpora 2012-2014[3] for experiments.

**ENTITY TRACK** The Entity track[4] ran consecutively from 2009 to 2011 and it consisted of two main tasks. The first task, Related Entity Finding (REF), was based on the ClueWeb09[5] dataset, as well as the Billion Triples Challenge datasets (BTC-2009[6] and BTC-2010[7]). The second task, Entity List Completion (ELC), was also based on the BTC-2009 and BTC-2010 linked open data. The overall problem tackled by the Entity track consisted of taking one (REF) or several (ELC) entities as a query and returning a ranked list of related entities.

**QUESTION ANSWERING TRACK** The Question Answering track[8] is one of the longest running tracks organized by TREC, starting in 1999 and running until 2007. It was based on several static datasets shared by other tracks, such as TREC disks 4&5[9], together with a set of test questions, either manually created, or taken from search logs donated by Microsoft or AOL. The track was inactive for eight years, until 2015, when it was revived as the Live QA track. The Live QA track also ran in 2016 and in 2017. The Live QA track was different from its precursor in the sense that it is required answers to be found for questions submitted to Yahoo Answers and pushed to participants as a data stream.

1 http://trec-kba.org/
2 http://trec-kba.org/data/2014-11-19-TREC-KBA-track-overview.pptx
3 http://s3.amazonaws.com/aws-publicdatasets/trec/kba/index.html
4 https://web.archive.org/web/20110811014305/http://ilps.science.uva.nl/trec-entity/
5 https://lemurproject.org/clueweb09/
6 https://km.aifb.kit.edu/projects/btc-2009/
7 https://km.aifb.kit.edu/projects/btc-2010/
8 http://trec.nist.gov/data/qamain.html
9 http://trec.nist.gov/data/docs_eng.html

**OPEN SEARCH TRACK** While most tracks provide a golden collection, with documents, topics and manually annotated relevance judgments, the OpenSearch track tackles the problem from a different angle. Participants are equally provided with documents and topics, but the assessment is done in a real-world scenario, via team-draft interleaving [245], an approach that combines the site's search results with the search results provided by the participant. Evaluation is then done based on the implicit feedback given by clicked results, accounting for the fraction of wins of the participant over the site (a result of 0.5 would represent an equivalent approach, while a higher result would represent a better approach). This is done through the Living Labs API[1], where users are given access to documents from well-known academic search engines, like CiteSeerX or Microsoft Academic Search. These documents must be indexed, using whatever strategy the participant chooses. Then, a list of frequently issued queries is provided to the participants, who must run them through their retrieval model for ranking and then submit the results to the Living Labs platform. This infrastructure is integrated with the real-world search engines that initially provided the documents. The search engines dedicate a small part of their traffic to evaluating runs from participants. Whenever a real user issues one of the supported queries, participant's results are interleaved with the search engine's results and shown to the final user. Feedback is sent back to the participant with information on which results (theirs or the search engine's) were clicked. This implicit feedback can then be used to evaluate the system or even to train a learning-to-rank approach, during a train stage of the track.

**Living Labs** Balog et al. [246] presented the first practical methodology and implementation of the Living Labs[2] for IR benchmarking, focusing on two use cases: local domain search on the website of the University of Amsterdam, and product search in the webshop of a toy retailer operating in Hungary. Living labs represents a central and shared experimental environment that replaces individually setup evaluation infrastructure. This infrastructure can be used by different research groups, avoiding the hassle of preparing it themselves, and enabling them to compare evaluation metrics within a similar platform. This is different from the classical approach of evaluating retrieval models through test collections and instead takes advantage of interleaving in real-world search engines. Therefore, it confers the approach a unique degree of feedback, which is frequently only available to commercial or institutional search engines, due to the challenges of finding a representative set of test subjects in a lab environment.

Table 2.4 provides a summary of the most relevant TREC tracks, their datasets and mission, in the context of entity-oriented search. One way to look at KBA is to interpret it as document ranking for an entity query, while the entity track can be seen as entity ranking for a query with one or multiple entities. The QA tracks, however, focus on query and document understanding, using this to improve the matching to specific passages. This is usually done through syntactic parsing and semantic tagging, identifying dependencies between parts of speech that include entities. Finally, the Open Search track illustrates a problem where a keyword query is used to retrieve a specific type of entities (publications), usually providing entity rich queries (e.g., authors, subjects). There are regularities in these retrieval tasks, which usually involve a text part (documents or passages), an entity part (in queries or results), and the inter and intra links between them. This poses the question and establishes the motivation to whether a general model can be found that is capable of solving a large number of these tasks using a universal approach. This is part of what we explore in this thesis with graph-based entity-oriented search.

1 http://doc.trec-open-search.org/en/latest/api-participant.html
2 https://bitbucket.org/living-labs/ll-api

Table 2.4: Overview of TREC tracks relevant to entity-oriented search.

| Track | Datasets | Mission |
|---|---|---|
| Knowledge Base Acceleration track[1] | TREC KBA Stream Corpora 2012-2014[3] | Improve the retrieval of documents about a particular entity from a stream, in order to accelerate manual knowledge base population. |
| Entity track | ClueWeb09[5], BTC-2009[6], BTC-2010[6] | Improve the retrieval of related entities based on one or multiple entities. |
| Question Answering track[8] | TIPSTER[1], TREC disks[9], MSN Search logs, AskJeeves logs, The AQUAINT Corpus of English News Text[2] | Improve the retrieval of document passages capable of directly answering user questions. |
| Live QA track[3] | Yahoo! Answers[4] | Similar to the Question Answering track, but questions are received directly from a socket, as a stream, in real-time. |
| Open Search track[5] | CiteSeerX[6], Microsoft Academic Search[7], Social Science Open Access Repository[8] | The Academic Search Edition consists of improving literature retrieval for a given a keyword query, which might include entities (e.g., authors). |

#### 2.3.3.2 *INEX – Initiative for the Evaluation of XML retrieval*

The INitiative for the Evaluation of XML Retrieval (INEX) began in 2002 aiming its attention at focused retrieval, over test collections provided in XML. The goal was to improve on directly solving the user's information needs. Instead of simply providing a ranking of documents, in focused retrieval we also rank passages or entities that could more directly answer user questions. INEX followed the same configuration of TREC, organizing into tracks for different focused retrieval problems, many of them relying on semantically annotated collections of documents, as well as datasets of linked open data. While some of the tracks relied on classical evaluation metrics like MAP (e.g., Ad Hoc track), others relied on variations of this metric. This included xinfAP (eXtended INFerred Average Precision), where the average precision was inferred based on a sample of relevant documents from the collection, selected using stratified sampling over different ranking positions. It also included MAiP (Mean Average interpolated Precision), which relied in the interpolated precision over different intervals of recall. Like TREC, INEX also provides an overview of each track's occurrence, with the evaluation metrics for the runs submitted by participants (e.g., Demartini et al. [247], for the 2009 Entity Ranking track). Next, we present an overview of the tracks that we have identified as relevant for entity-oriented search in Table 2.3.

**AD HOC TRACK** The Ad Hoc track ran from 2007 to 2010, with the goal of exploring the internal structure of documents to retrieve relevant information, usually as individual passages, or sometimes as a way to improve the retrieval of entire documents (document-level qrel were provided for the 2010 occurrence). Three main tasks were maintained throughout all occurrences of the Ad Hoc track: (i) the Focused task, where a ranked list of XML elements or passages was returned; (ii) the Relevant in Context task, where non-overlapping elements or passages were returned, grouped by their article of origin; and (iii) the Best in Context task, where a single element or passage was returned along with its article of origin. In 2009, the Thorough task was revived and introduced into the Ad Hoc track and, in 2010, the Efficiency task was introduced, along with modifications to the original tasks, introducing restrictions based on a maximum number of characters per article, or per topic, affecting the returned results. In 2007 and 2008, runs were based on a XML version of the English Wikipedia, with over 500k articles from the beginning of 2006. In 2009 and 2010, a new XML dataset was used instead, with over 2.5M Wikipedia articles from 2009, that were semantically annotated based on the YAGO ontology. Using as reference the last occurrence in 2010, as well as article retrieval

**Table 2.5:** Overview of INEX tracks relevant to entity-oriented search.

| Track | Datasets | Mission |
|---|---|---|
| Ad Hoc track | 2006 English Wikipedia (XML), INEX 2009 Wikipedia collection (XML, annotated with YAGO classes) | Explore the internal structure of documents to retrieve relevant information. |
| Entity Ranking track | 2006 English Wikipedia (XML), INEX 2009 Wikipedia collection (XML, annotated with YAGO classes) | Explore the direct retrieval of entities as a way to solve information needs, either based on a keyword query or a multiple entity query. |
| Linked Data track | Wikipedia-LOD v1.1, Wikipedia-LOD v2.0 | Explore retrieval techniques over a combination of text and linked data. |

with document-level relevance [248, Table 13], we found values of MAP ranging from 0.3177 to 0.4294.

**ENTITY RANKING TRACK** The Entity Ranking track ran from 2007 to 2009, offering two different tasks to their participants: (i) the Entity Ranking task, where the query was expressed in natural language; and (ii) the List Completion task, where the query was expressed as a set of example entities. Both tasks returned a list of ranked entities, based on either a text query or an entity query. This track relied on the same Wikipedia collections as the Ad Hoc tracks occurring in the corresponding years. This is an advantage for exploring general retrieval models, since a universal ranking function could rely on an indexing of the same collection to be assessed. Using as reference the last occurrence in 2009, we found values of xinfAP ranging between 0.082 and 0.517 for the Entity Ranking task, and between 0.100 and 0.520 for the List Completion task.

**LINKED DATA TRACK** The Linked Data track ran in 2012 and 2013. It focused on combining the information from textual Wikipedia articles structured as XML, with associated semantic relations from RDF properties based on DBpedia and YAGO2. Three tasks were considered for the Linked Data track: (i) the Ad Hoc Retrieval task, similar to the prior Ad Hoc track, but using leaf topics from a three-level hierarchy generated based on random topics using Google suggestions; (ii) the Faceted Search task, which also relied on the same three-level hierarchy using the non-leaf topics; and (iii) the Jeopardy task, where topics provided a clue, a keyword query, and a SPARQL query proposing a full-text search operator *FTContains* that was not supported by SPARQL. For the Jeopardy task, participants were motivated to propose solutions for indexing RDF and textual content, a goal that we also explore in this thesis by proposing graph-based general representation models. Using as reference the last occurrence in 2013, we found values of MAiP ranging between 0.1302 and 0.3128 for the Ad Hoc track over the Jeopardy topics, and between 0.7010 and 0.7491 for the Jeopardy task.

Table 2.5 provides a summary of the most relevant INEX tracks, their datasets and mission, in the context of entity-oriented search. In this doctoral work, we focus on proposing a general representation and retrieval model for entity-oriented search tasks. As such, it is of special interest to use a common test collection that provides relevance judgments for multiple tasks. This is the case of the datasets used in the Ad Hoc and Entity Ranking tracks. In particular, the INEX 2009 Wikipedia collection is used in both tracks, providing relevance judgments for three entity-oriented search tasks, while being easily accessible online.

**Table 2.6**: Characterizing the entity tracks in TREC and INEX by their input and output.

| Track | Task | Input: Keywords | Input: Source entity | Input: Example entities | Input: Target entity type | Input: Relation | Output (entities): Homepages | Output (entities): Linked Data URIs | Output (entities): Wikipedia pages |
|---|---|---|---|---|---|---|---|---|---|
| TREC | REF | | ✓ | | ✓ | ✓ | ✓ | | |
| | ELC | | ✓ | ✓ | ✓ | ✓ | | ✓ | |
| INEX | ER | ✓ | | | ✓ | | | | ✓ |
| | LC | ✓ | | ✓ | | | | ✓ | |

#### 2.3.3.3 *Comparing the entity tracks in TREC and INEX*

The entity tracks from both TREC and INEX contain similar tasks, that rely on a very similar set of input and output elements. We analyze the definition of each task, as provided by its first occurrence in the respective overview documents.

Balog et al. [70] described the Related Entity Finding task, from the TREC Entity track, as follows:

> *Given an input entity, by its name and homepage, the type of the target entity, as well as the nature of their relation, described in free text, find related entities that are of target type, standing in the required relation to the input entity.*

In the 2011 occurrence, linked open data URIs were used to replace homepages in the representation of input entities. In 2010, the Entity List Completion task, from the TREC Entity track, was also introduced [69] and defined as follows:

> *Given an information need and a list of known relevant entity homepages, return a list of relevant entity URIs from a specific collection of Linked Open Data.*

INEX also provided two tasks in their Entity Ranking track. In 2007, Vries et al. [68] introduced the Entity Ranking task, defining it as follows:

> *The motivation for the Entity Ranking task is to return entities that satisfy a topic described in natural language text. [...] An Entity Ranking topic specifies the category identifier and the free-text query specification. Results consist of a list of Wikipedia pages [...].*

Also in 2007, they introduced the List Completion task, from the INEX Entity track, defining it as follows:

> *In the List Completion task, instead of knowing the desired category (entity type), the topic specifies between one and three correct entities (instances) together with a free-text context description. Results consist again of a list of entities (Wikipedia pages).*

Table 2.6 organizes the definitions of the previous four tasks, from TREC and INEX, in a comprehensive manner. We identify the input and output elements so that it becomes clear what each task has in common. This information is particularly helpful in distinguishing between the TREC and INEX versions of the entity list completion task, but also in understanding the commonalities between all tasks. We consider such approach to be a fundamental step when designing a general model in any area of information retrieval. As expected, the output for all task is a ranking of entities, although different entity representations are used, either relying on their homepages, a URI identifying the entity in a knowledge graph, or a Wikipedia page about the entity. Regarding input, both TREC tasks rely on a source entity, a target entity type and a relation, while both INEX tasks rely on a keyword query instead. Both entity list completion tasks (TREC ELC and INEX LC) take advantage of example entities that can be used as relevance feedback to further specify the information need. Almost all tasks rely on a target entity type, with the

exception of INEX LC that only relies on the example entities, as an alternative to the target entity type. Despite the different inputs that each task takes, they have a lot in common, if we increase the abstraction level, as both attempt to reach and rank entities, either through free-text or an entity identifier, providing a greater or lesser level of restrictions through examples, target types or relations.

## 2.4 DISCUSSION

In this section, we present several observations, identifying possible trails leading to the future of graph-based entity-oriented search. We end the section with an overview on the overall classes of graph-based models presented in this survey.

### 2.4.1 Observations

We present several remarks surrounding graph-based entity-oriented search, its relation to semantic search and the exploration of higher-order dependencies with hypergraphs, proposing future directions towards hypergraph-based quantum search[1].

#### *The relation between entity-oriented search and semantic search*

One particular source of confusion is the definition of semantic search and how it relates to entity-oriented search. Most of the work we reviewed either refers to semantic search as document retrieval leveraging entities, or as entity retrieval over linked data. In its broader definition [62, Def.1.6], semantic search subsumes entity-oriented search. However, when considering any of the described tasks, we might say that semantic search is instead subsumed by entity-oriented search. In practice, detaching the semantic search classification from any specific task might be the most adequate approach, thus promoting the use of the broader and more abstract definition, and instead more clearly describing the tasks as ad hoc document retrieval and ad hoc entity retrieval, respectively. In this survey, we complied with the definitions proposed by Balog [62], except when the cited paper specifically mentioned a semantic search task, in which case we clearly stated which definition the authors adhered to.

#### *What is and isn't a graph-based model?*

We defined graph-based models as any approach that relied on a graph, at whichever stage of the process. This included graphs for representing:

- Text (e.g., linking terms within a window, or with similar embeddings);
- Entities, their attributes and relations (i.e., knowledge graphs);
- Relations between documents (e.g., hyperlinks, similarity).

While many probabilistic models might also be considered graph-based, namely Bayesian or Markov networks, we opted to classify them as probabilistic, unless they were clearly operating over a specific graph (e.g., web graph, similarity graph). PageRank might be the most evident example of a probabilistic graph-based model, since it is clearly applied to the web graph to rank web pages by importance. This is why it was relevant to cover overall probabilistic models, before delving into graph-based models.

1 Please note that quantum approaches to information retrieval have already been explored in the past, for instance with the quantum language models by Sordoni et al. [249]

As an area, graph-based entity-oriented search still has a lot of unexploited potential, in particular regarding approaches developed in network science. This includes PageRank, which as been abundantly used, but also other centrality metrics like closeness or betweenness, as well as community detection or motif discovery. Graph connectivity can be studied from three main perspectives: microscale (node or edge properties), mesoscale (community or motif level) and macroscale (global). There are still many unexplored approaches, at all scales, that might be useful to better understand information in the context of search (e.g., graphlet-orbit transitions as a way to establish graph similarity [250]).

#### *From binary dependencies to higher-order dependencies*

A current trend in machine learning is the application of tensors for representing higher-order dependencies, particularly popularized by Google's TensorFlow [251]. Similarly, hypergraphs are able to elevate the expressiveness of a graph's binary dependencies to higher-order dependencies. In Section 2.2.7, we have seen that there some hypergraph-based approaches for indexing, representing and querying documents. However, there hasn't been much work specifically directed at entity-oriented search. We argue that further exploring hypergraphs, without falling back to the domain of graphs, might lead to useful and novel strategies to better solve information needs. A possible approach is the application of PageRank to "knowledge hypergraphs", where a random surfer would, at each step, randomly select a hyperedge and then randomly select a node from that hyperedge [55]. As the complexity of the hypergraph increases, particularly for memory-based hypergraphs (i.e., that explicitly store information statements), even random walk based approaches become inefficient for real-time computation. However, we know that random walks in graphs can be modeled using Markov chains, which are stochastic models whose simulation is being studied in quantum computer [252]. In turn, implementing random walks in hypergraphs using a quantum computer would also require a Markov process to be defined over a hypergraph [253]. We also argue that, for this reason, the complexity of such models and the overall predicted inefficiency should not be reasons to discard it as a viable approach, worthy of study. It is this holistic view, leading to general models for information retrieval, based on hypergraphs, that we introduce in this thesis.

### 2.4.2 An overview on entity-oriented search approaches

Entity-oriented search is a naturally heterogeneous area, where documents and entities are combined to better solve the information needs of the users. When querying, users can take advantage of keyword or natural language queries, as well as entity queries, obtaining results that can either include documents, entities, or both. However, techniques for document and entity representation have been quite disjoint, with the inverted index taking the lead to represent multi-field documents, and the triplestore taking the lead to represent entities, their types, attributes and relations. Some of the first approaches to tackling entity-oriented search tasks, were based on translating the problem to the domain of classical information retrieval. Please refer to Table C.1 for an overview of these approaches, based on virtual documents, combined data and probabilistic graphical models. Other approaches integrated information from documents and entities based on learning to rank models. That way, signals from different representations (e.g., inverted index and triplestore) could be combined based on a learned ranking function, trained for instance using a support-vector machine or a neural network. Table C.2 can be used as a reference for the learning to rank models that we covered, illustrating semantic-driven, virtual document, and representation learning approaches.

Graph-based models can also be used as a way to integrate information from documents and entities, harnessing techniques developed through years of research

on information retrieval and network science (e.g., PageRank [113]), as well as graph-based representations developed individually for either type of data (e.g., graph-of-word [16] for documents and RDF[1] for entities). While graphs have been prevalent in information retrieval, using them to solve the representation mismatch between documents and entities is fairly recent. Table C.3 provides a reference for graph-based approaches, both general and specific to entity-oriented search. In particular, we covered general link analysis approaches. We also covered text as a graph, which, despite no entities being considered, provided a common ground for integration with entity graphs. We then covered knowledge graphs and how they are built and used for document and/or entity search. We examined text to entity graph approaches, where information extraction was used to acquire a structured graph to represent the document by its entities and relations. We then covered graph matching approaches, where a query graph is matched against subgraphs in an entity graph. Finally, we considered hypergraph-based models, with potential applications to entity-oriented search, and we closed with random walk based models, that are based on PageRank adaptions to an entity-oriented context.

Table 2.7 provides a comprehensive view of the surveyed approaches for each of the three models — classical IR, learning to rank, and graph-based models — along with the tasks that they support.

The four main entity-oriented retrieval tasks that we considered were:

- Ad hoc document retrieval (leveraging entities);
- Ad hoc entity retrieval;
- Related entity finding;
- Entity list completion.

Other related entity-oriented and semantic retrieval tasks included:

- Sentence retrieval;
- Answer tree ranking;
- Attribute retrieval;
- Relation retrieval.

Knowledge graph related tasks included:

- Knowledge graph construction and modeling;
- Node importance;
- Node relatedness;
- Graph partitioning.

And we also included:

- Topic modeling;
- Text classification;
- Joint representation;
- Document representation.

As we can see, the task with the highest coverage was ad hoc entity retrieval. Combined data approaches are able to support all of the four main entity-oriented search tasks that we considered. The reviewed graph matching and random walk based approaches are able to support three out of the four tasks, with ad hoc document retrieval missing. However, graph-based models were used to represent text as a graph, to structure knowledge bases, to convert text to an entity graph, and in hypergraph-based approaches for ad hoc document retrieval. This supports our thesis that the graph data structure might be viable as a joint representation model, able to support the four retrieval tasks, and providing a framework to develop a universal ranking function. One example of a basis for such a function would be the heat kernel PageRank, which is able to measure node importance and node relatedness, as well as a to obtain a graph partition.

1 https://www.w3.org/RDF/

**Table 2.7**: Categorization of entity-oriented search approaches and applications.

| | Category | References | Ad Hoc Document Retrieval | Ad Hoc Entity Retrieval | Related Entity Finding | Entity List Completion | Sentence Retrieval | Answer Tree Ranking | Attribute Retrieval | Relation Retrieval | Knowledge Graph Construction and Modeling | Node Importance | Node Relatedness | Graph Partitioning | Topic Modeling | Text Classification | Joint Representation | Document Representation |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Classical** | Virtual Documents | [3, 4, 132] | | ✓ | | | | | ✓ | ✓ | | | | | | | | |
| | Combined Data | [74, 123, 128, 130] | ✓ | ✓ | ✓ | ✓ | | | | | | | | | | | ✓ | |
| | Probabilistic Graphical Models | [124–126] | | ✓ | | | ✓ | | | | | | | | | | | |
| | Cluster Hypothesis | [73] | ✓ | | | | | | | | | | | | | | | |
| **Learning to Rank** | Semantic-Driven | [137, 141, 143] | | ✓ | ✓ | | ✓ | | | | | | | | | | | |
| | Virtual Documents | [144] | | ✓ | | | | | | | | | | | | | | |
| | Representation Learning | [86] | | ✓ | | | | | | | | | | | | | | |
| **Graph-Based** | Link Analysis | [113, 149, 151, 162–165, 169] | | | | | | | | | | ✓ | ✓ | ✓ | | | | |
| | Text as a Graph | [15, 16, 171] | ✓ | | | | | | | | | | | | | ✓ | | ✓ |
| | Knowledge Graphs | [54, 172, 174–176, 179, 185, 186, 254] | ✓ | ✓ | | | | | | | ✓ | | | | ✓ | | ✓ | |
| | Text to Entity Graph | [187, 189] | ✓ | ✓ | | | | | | | | | | | | | | |
| | Entity Graph to Tensor | [75] | | ✓ | | | | | | | | | | | | | | |
| | Graph Matching | [57, 58, 196, 198, 199] | | ✓ | ✓ | ✓ | | ✓ | | | | | | | | | | |
| | Hypergraph-Based | [14, 76, 103, 203, 204] | ✓ | ✓ | | | | | | | | | | | | | ✓ | ✓ |
| | Random Walk Based | [65–67, 72, 238, 239] | | ✓ | ✓ | ✓ | | | | | | ✓ | | | | | | |

### 2.4.3 An automated analysis of the bibliography

We carried an automated analysis of the literature collected during the state of the art survey process. We prepared a Jupyter notebook based on the R kernel[1] to derive several statistics and plots from the BibTeX file included in this dissertation, and from several entries added to the doctoral wiki about a subset of selected publications (see Section 3.2 for further details on this systematic documentation approach).

We first built two CSV files, one by parsing the BibTeX file and another one by scraping the doctoral wiki pages within the *phd:bibliography* namespace. Each generated CSV file contains the following fields, which are empty when unavailable: *title*; *author* (multiple authors were separated by '|'); *year*; *conference*; *core* (conference rank based on CORE 2018 data); *journal*; *scimago_h_index*, *scimago_sjr* and *scimago_quartile* (journal h-index, SJR and SJR quartile, as extracted from SCImago 2018 data); *institution*; *publisher*; and *review* (only available for some of publications in the doctoral wiki).

In the BibTeX file, we used simple heuristics based on common expressions to obtain the conference name from the *booktitle* field corresponding to the proceedings entry. We then used the Jaccard index to match the conference name with an entry in CORE 2018, in an attempt to normalize it. We also used this approach to obtain the core ranking and the SJR quartile for all conference and journal entries from both BibTeX and wiki bibliographical data. In the following two sections, we present the results that we obtained.

#### 2.4.3.1 *BibTeX based statistics*

According to Table 2.8, we our analysis of the BibTeX file covers 499 publications, with an average of 2.28 authors, out of 1,140 distinct authors. Publications range from 1847 to 2019, covering 182 conferences with ranks between A∗ and C, as well as 123 journals with SJR from 0.131 to 16.345 and h-indexes from 12 to 1,096, 8

1 https://irkernel.github.io/

**Table 2.8:** BibTeX: Publication statistics.

| | |
|---|---|
| **Number of Publications** | 499 |
| **Number of Authors** | 1,140 |
| **Authors per Publications** | 2.28 |
| **Year Coverage** | 1847–2019 |
| **Number of Conferences** | 182 |
| **CORE 2018 Coverage** | A*–C |
| **Number of Journals** | 123 |
| **SCImago SJR** | 0.131–16.345 |
| **SCImago H-index Range** | 12–1,096 |
| **Number of Institutions** | 8 |
| **Number of Publishers** | 42 |

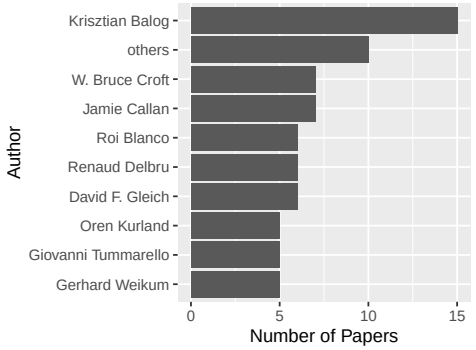



**Figure 2.1:** BibTeX: Top 10 most cited authors.

institutions (usually universities associated with master or doctoral theses), and 42 publishers.

Figure 2.1 shows the number of papers for the top 10 most cited authors (ignoring repeated citations per paper). In that list, we can identify several well-known authors from relevant areas to this thesis. Among them, we find Krisztian Balog, who has been distinguished with the Karen Spärck Jones award for his scientific contributions to information retrieval, which were greatly focused on entity-oriented search [62] and evaluation [246]. We also find W. Bruce Croft who has put forward several innovative and central ideas like language models for information retrieval [35], a reflection on the similarities of information filtering and information retrieval [87], or the modeling of term dependencies through Markov random fields [120] or query hypergraphs [14]. Jamie Callan was considered for his contributions with the Lemur toolkit [255] and INQUERY retrieval system [256], to which W. Bruce Croft also contributed. Callan has also more recently worked on with Chenyan Xiong, developing work on joint representation models for words and entities [257, 258], as well as for entity-oriented search using learning to rank [144], even contributing with a test collection based on DBpedia [184]. Roi Blanco has contributed with the groundwork for graph-based information retrieval, proposing several approaches for modeling documents as graphs and computing term weights from this representation [15]. David F. Gleich has done strong contributions in the area of PageRank, both providing an excellent survey on the subject [155], and proposing the Multilinear PageRank [114] as a higher-dimension generalization of PageRank applicable to tensors. Renaud Delbru has also contributed with PageRank approaches, applied to entity-oriented search, or more specifically the web of data [72, 259]. He was also one of the creators, along with Balog and Tummarello (also on the top 10), of the Sindice Dataset [244], an historical test collection from entity-oriented search that is no longer available sensibly since the creation of the SindiceTech startup company. Oren Kurland explored the cluster hypothesis for entity-oriented search [73], as well as document retrieval using entity-based language models [260], or Markov random fields applied to entity-oriented search [124]. He also explored PageRank with language models [261]. Finally, Gerhard Weikum was involved in the creation of the YAGO ontology [95], as well entity disambiguation tasks [262, 263] and RDF graph querying approaches based on language models [264].

Figure 2.2 shows the distribution of cited literature over the publication years (left), as well as the top most cited years (right). As we can see, there are a few outliers that include citations from the 1800s, however research is mostly focused on the first and second decade of the 2000s, with 2012 and 2007 being the top two cited years, covering work about graph-based approaches, entity-oriented search and the semantic web.

Figure 2.3 shows the top 10 most cited conferences along with the distribution of all ranked conferences according to CORE 2018. As we can see, the top 3 confer-

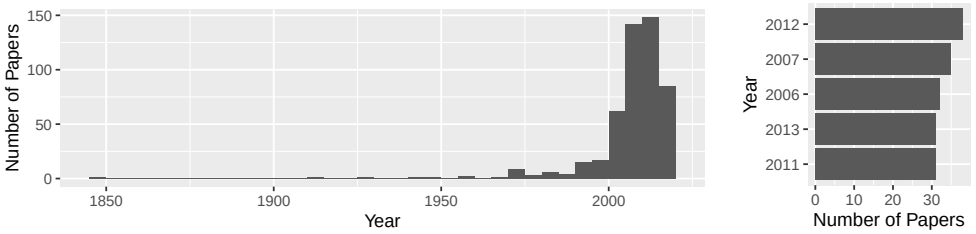



**Figure 2.2:** BibTeX: Publications distribution per year (top 5 years on the right).

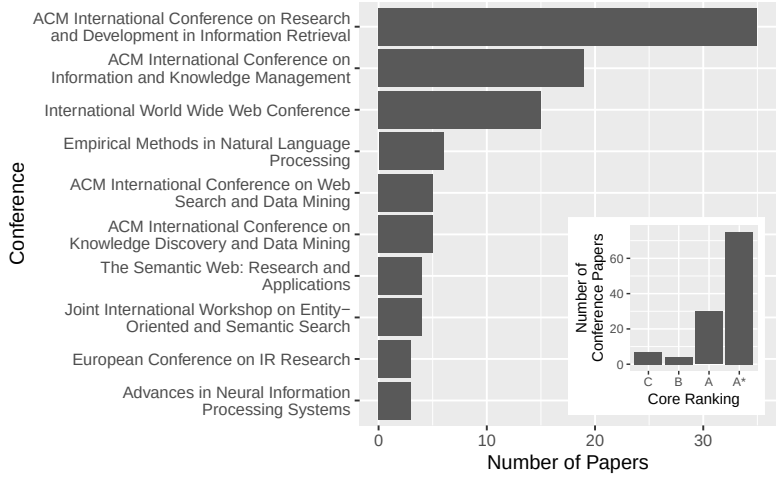



**Figure 2.3:** BibTeX: Top 10 most cited conferences (CORE 2018 rank distribution on the right).

ences included SIGIR (conf.), which is the main conference in information retrieval, with A∗ rank, CIKM, which is also a relevant conference, with A rank, that covers information retrieval as well as knowledge management topics, both central to this doctoral work, and WWW, with A∗ rank, which covers topics like link analysis or semantic web aspects like RDF, microformats or the YAGO and Wikidata knowledge bases. As we can see on the core ranking distribution, most of the cited work is from conferences with A∗ and A ranking, with only a few B and C rank conferences.

Figure 2.4 shows the top 10 most cited journals along with the SJR (SCImago Journal Rank) quartile distribution of all ranked journals according to SCImago 2018 data. As we can see, the most cited journal is the non-peer-reviewed Computing Research Repository (CoRR) from arXiv. During our literature review process, we prioritized the peer-reviewed versions of archived preprints, when available. Even after carefully reviewing the BibTeX to account for this, the number of considered citations from CoRR was still high. This is, in part, due to the fact that some of the approaches that we explore in this thesis, namely based on hypergraphs, are still quite novel and only now beginning to trend [105, 115, 201, 205]. In second place we find ACM Transactions on Information Systems, with work on signature files, a classical indexing model [26], or graph-based ranking approaches [158, 196]. Tied in third and fourth places, we find Foundations and Trends in Information Retrieval, along with Communications of the ACM. In fifth place, we find Internet Mathematics, because its in-depth work on PageRank. The remaining top journals cover the information science, knowledge management, the semantic web, and information systems. As we can see from the SJR quartile distribution, most of the cited journals are in the first quartile (Q1), progressively including fewer journals as the quartile moves from Q2 to Q4. This is the expected behavior, as higher quality content is usually present in higher ranked journals.

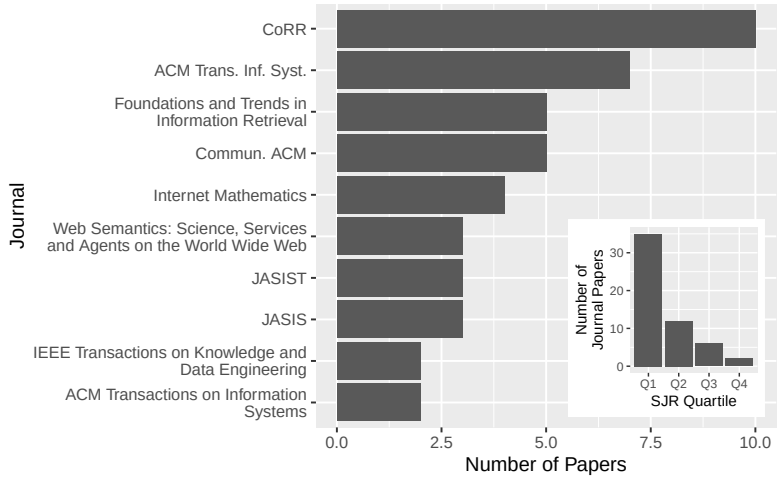



**Figure 2.4:** BibTeX: Top 10 most cited journals (SJR quartile distribution on the right).

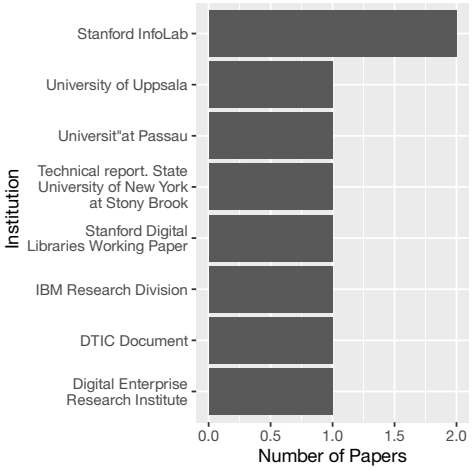



**Figure 2.5:** BibTeX: Top 10 institutions.

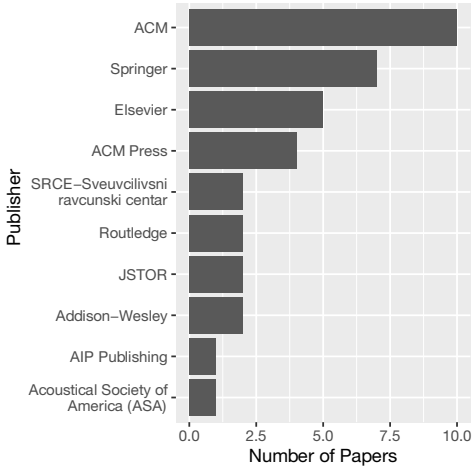



**Figure 2.6:** BibTeX: Top 10 publishers.

Figure 2.5 shows the 8 institutions mentioned in citations, usually regarding master or doctoral theses, or technical reports. As we can see, we cited Stanford InfoLab twice for their technical reports on the original PageRank [113] and approaches for its efficient computation [229]. Figure 2.6 shows the top 10 cited publishers, with ACM, Springer and Elsevier, as the most cited publishers, which is expected, given their prevalence in computer science collections.

Finally, we also looked at the most frequent terms (unigrams and bigrams) used in the titles of the cited publications, without highly common stopwords. While most of the terms are quite predictable (e.g., *search*, *entity*, *information retrieval*, or *semantic*), there are also some terms that are indicative of the particular interests in this thesis, like *pagerank*, *graph*, or *hypergraphs*. While for this thesis, the analysis we have presented so far is enough to illustrate the explored literature, it would be interesting to extend this work, for instance considering a better approach than term frequency to measure term importance in publication titles. Perhaps maximal k-core extraction using graph-of-word would be and interesting approach [111].

#### 2.4.3.2 *Doctoral wiki based statistics*

We used a DokuWiki instance for a systematic documentation of the work developed throughout this thesis (see Section 3.2 for further details). A part of this doctoral wiki was dedicated to keeping reading sheets for reviewed literature. In

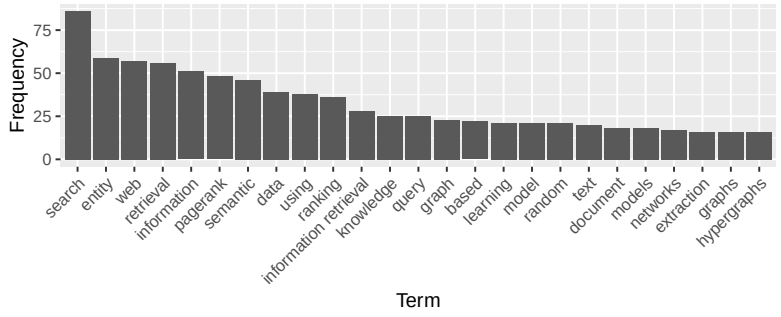



**Figure 2.7:** BibTeX: Top 25 most frequent unigrams and bigrams in titles.

**Table 2.9:** Wiki: Publication statistics.

| | |
|---|---|
| **Number of Publications** | 119 |
| **Number of Authors** | 375 |
| **Authors per Publications** | 3.15 |
| **Year Coverage** | 1957–2019 |
| **Number of Conferences** | 28 |
| **CORE 2018 Coverage** | A*–C |
| **Number of Journals** | 30 |
| **SCImago SJR** | 0.146–3.658 |
| **SCImago H-index Range** | 12–180 |
| **Number of Institutions** | 14 |
| **Number of Publishers** | 3 |

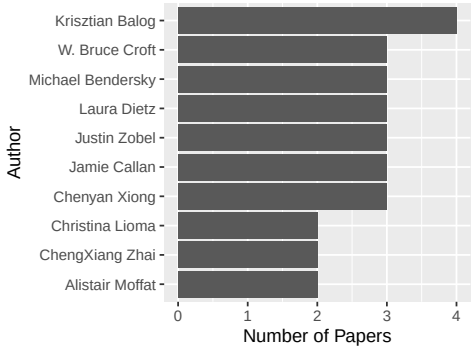



**Figure 2.8:** Wiki: Top 10 most cited authors.

this section, we repeat the analysis carried over the BibTeX file, this time using wiki information as the data source. Literature covered in the wiki represents a focused selection of central contributions, that usually cover topics that are closest to this thesis. This means they are different from those in the BibTeX, greatly overlapping, almost as a small subset, but also containing unused work collected during the initial stage of exploration. Additionally, the wiki does not include most of the well-known references from information retrieval, since these were not very often examined in detail due to having been already explored in the past, or continuously consulted throughout this work. Our analysis focuses particularly on the bibliographic differences between the BibTeX and the wiki, and it reflects the interest and relevance assigned to the selected publications.

As we can see in Table 2.9, the doctoral wiki contains only 24% of the publications in the BibTeX, which corresponds to 119 reading sheets. The number of considered authors was reduced to 375, but the average number of authors per publications slightly increased to 3.15. The oldest publication added to the wiki was from 1957, corresponding to Luhn's work that led to first statistical approaches for information retrieval [8]. In the wiki, we covered 28 conferences, from ranks A$*$ to C, 30 journals, with SJR ranging from 0.146 to 3.658 and h-index ranging from 12 to 180, 14 institutions, and 3 publishers. The increased number of institutions in the wiki, when compared to the BibTeX, can be explained by the fact that not all reviewed literature was necessarily cited.

Figure 2.8 shows the top 10 most cited authors in the wiki. When compared to the BibTeX, there are three authors in common, Krisztian Balog, W. Bruce Croft, and Jamie Callan, which further reinforces their relevance. Additionally, we find other central authors to the topic of this thesis, namely Michael Bendersky, who proposed the query hypergraph model, Laura Dietz, who proposed the ENT Rank model, also based on a hypergraph, or Chenyan Xiong, who focused on joint representations for words and entities, and unified models for query entity linking and ad hoc document retrieval (leveraging entities).

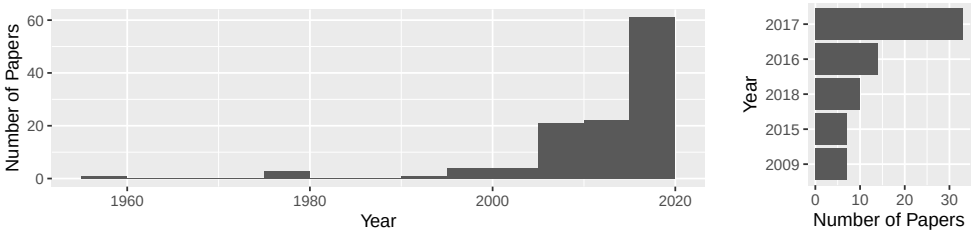



**Figure 2.9:** Wiki: Publications distribution per year (top 5 years on the right).

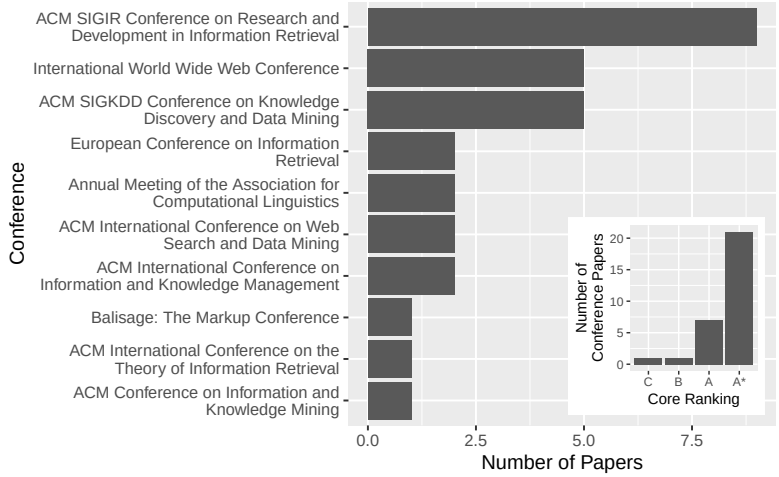



**Figure 2.10:** Wiki: Top 10 most cited conferences (CORE 2018 rank distribution on the right).

Figure 2.9 shows a similar behavior for the yearly distribution of publications in the wiki, when compared to the BibTeX. However, there is a clear bias towards recent content, with 2017 corresponding to the year with the highest number of publications.

Figure 2.10 illustrates the top 10 conferences corresponding to publications added to the doctoral wiki. We find similarities between the BibTeX and the doctoral wiki entries, with SIGIR (conf.) and WWW still in the top 3, but we also find KDD as a new entry to the top 3. The increased relevance of ECIR and ACL also become more evident, since they are now tied with CIKM and WSDM. We also notice the entrance of "Balisage: The Markup Conference" to the top 10, a slightly obscure and unranked conference, where we found interesting approach at modeling structured documents through a hypergraph [76]. Regarding the core ranking, we find a slightly stronger bias towards publications from A∗ conferences and a relative decrease of publications from C ranking conferences.

Figure 2.11 illustrates the top 10 journals corresponding to publications added to the doctoral wiki. There is a clear bias towards "Foundations and Trends in Information Retrieval", although in practice we only reviewed one of these publications [92], finding others interesting for future reference and review. Tied in the first position, we find arXiv, which is analogous to the CoRR entry from the BibTeX (Figure 2.4), since all articles are classified as Computer Science. However, this entry aggregates several publications that were later published in peer-reviewed conferences, and some of them were even renamed. While we updated the BibTeX to reflect this new information, the doctoral wiki only retained the information available at the date of review. Tied, in second place, we find JASIST (Journal of the Association for Information Science and Technology), which increased in relevance for wiki entries, along with the Information Retrieval Journal, and the ACM Computing Surveys. The SJR distribution based on SCImago data from 2018 shows a similar behavior to the BibTeX, with most of the journal publications being in the first quartile. On the

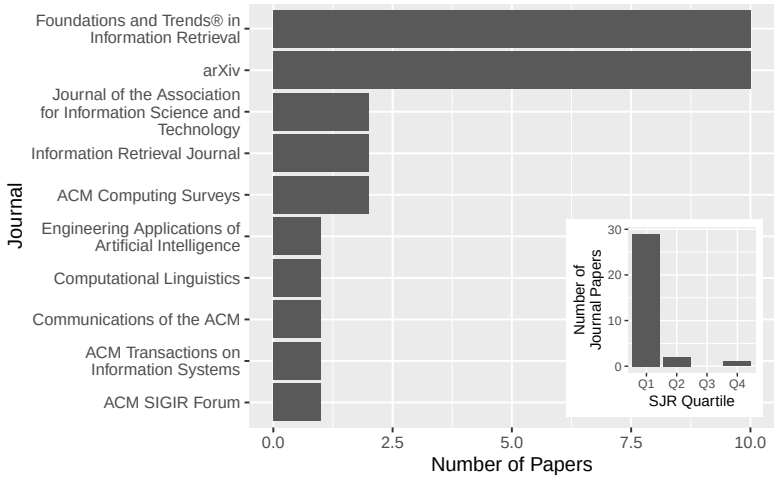



**Figure 2.11:** Wiki: Top 10 most cited journals (SJR quartile distribution on the right).

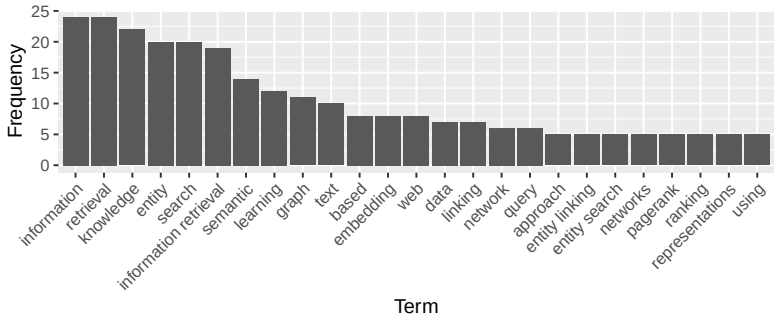



**Figure 2.12:** Wiki: Top 25 most frequent unigrams and bigrams in titles.

other hand, the number of journals from the remaining three quartiles is negligible, showing that only high quality publications were selected to be reviewed in the wiki.

Figure 2.12 shows the top 25 most frequent terms (unigrams and bigrams) used in the titles of articles with entries in the doctoral wiki. As we can see, they are quite similar with the previously analyzed terms for titles in the BibTeX file (cf. Figure 2.7), however top terms reflect more accurately the domain of research, including *information retrieval* at a higher (relative) frequency, as well as *knowledge*, *entity*, and *graph*. We also find that bigrams like *entity linking* and *entity search* have now entered the top 25, and *pagerank* has significantly decreased in rank according to frequency. Overall, we find that the keywords show a better alignment with the area of research, which is information retrieval, and particularly entity-oriented search.

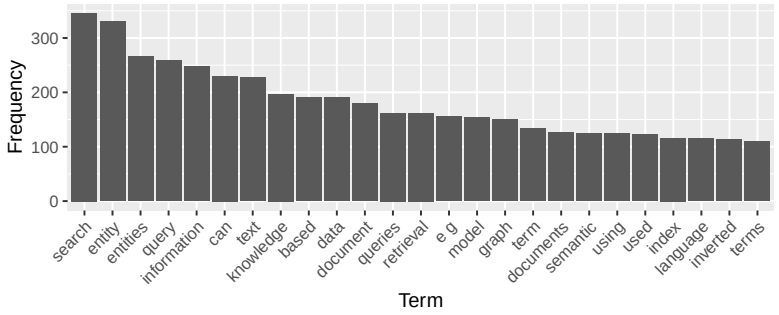



**Figure 2.13:** Wiki: Top 25 most frequent unigrams and bigrams in reviews.

Figure 2.13 illustrates the top 25 most frequent terms (unigrams and bigrams) used in the 20 reading sheets containing review notes. When comparing Figures 2.7 and 2.12, we find new terms that help us understand the literature on the topic, including *query*, *text*, *biased*, *document*, *index*, or *language*. Apart from these new terms, overall we find a similar distribution of terms, focused on *search*, *entity/entities*, and *knowledge*. We also include several examples in our notes, which we can confirm through the frequent usage of *e.g.*

### 2.4.4 A view of the future

While no explicit solutions have been put forward by the scientific community so far, the literature shows that graph-based approaches have the ability to support the definition of general models for information retrieval. We propose this could be done through a joint representation of units of information, and a universal ranking function reliant on that representation.

Looking at the future, graph-based models for entity-oriented search have the potential to represent both corpora and knowledge bases in a unified manner. When compared for instance with linear or log-linear models or learning to rank approaches, which can also consider a mix of signals from text and entities, graph-based models can go well beyond the integration of weights. That is, a well designed model should be able to harness the combined power of information within structured and unstructured data, at a low level of granularity, to better solve the user's information need. If properly abstracted as nodes or edges in a graph, text and entities become identical units of information, transforming into clues whose connections can be used to gather the elements that answer a user's question. Either way, not only can graphs provide a more unified approach to ranking, they can also be used to implement multiple retrieval tasks, without the need for a different combination of signals or a newly trained model, given the large number of individual tasks that can already be solved with similar graph-based models.

## SUMMARY

With the increasing relevance of entity-oriented search, it makes sense to look at graph-based models for information retrieval in a new light. We started this survey by providing context, presenting some historical perspective along with basic concepts and models from information retrieval. We covered general entity-oriented search approaches and general graph-based approaches, as well as a combination of both. Given the growing potential of the area, this survey focused on identifying a diverse set of representative methods, rather than doing an exhaustive research of all existing applications in each category. Our goal was to provide a map of opportunities in graph-based entity-oriented search, supporting not only this doctoral work, but also the future research on general models and universal ranking functions for information retrieval.

We surveyed the usage of classical information retrieval models, as well as learning to rank models, for entity-oriented search. Then, we provided a wide coverage on graph-based models, introducing classical link analysis approaches, like PageRank, HITS and kernel-based methods. We also described approaches for representing text as a graph, capturing discourse properties like context (e.g., graph-of-word). We described knowledge graph construction and modeling, along with its applications, either for improving ad hoc document retrieval or for supporting ad hoc entity retrieval. We studied approaches based on extracting entity graphs from text and using them as a complement for the representation and retrieval of documents. We also covered the usage of tensors to represent entity graphs and to obtain entity embeddings. We explored graph matching for querying with graphs — usually generated from natural language queries. We examined general hypergraph-based models for document representation, joint representation and ad hoc document retrieval, showing the potential for applications in entity-oriented search as well. We closed the graph-based section with random walk based models, mostly derived from PageRank and applied to entity graphs over a given context (e.g., web graph, dataset). We also provided a section on evaluation forums and datasets, useful for assessing a wide range of entity-oriented search tasks. Finally, we presented a discussion that covered several individual observations, providing an overview on entity-oriented search approaches, commenting on the results of an automated bibliographical analysis carried during the literature review process, and closing with a view of the future leading to the contributions in this thesis.

# Part II

# MATERIALS AND METHODS

# 3 RESEARCH METHODOLOGY

## Contents



The scientific method has been used since the 17th century. Information retrieval research follows this empirical cycle, frequently relying on community resources like evaluation forums, public test collections, or online evaluation approaches that integrate with real-world systems. Experiments either involve the measurement of effectiveness or efficiency, frequently focusing on a single one of these aspects. Methodology-wise, there is also a lot to be said about the documentation process of the research process itself, including reviewed literature, carried experiments, generated or exploited datasets, or even produced communications.

The structure of this chapter is organized as follows:

- **Section 3.1** covers the research methodology for information retrieval, based on the empirical cycle, that we follow in this thesis.
- **Section 3.2** provides an in-depth description of our systematic approach to documenting the doctoral work, based on a wiki content management system.

## 3.1 EMPIRICAL RESEARCH BASED ON TEST COLLECTIONS

Robertson [265] has described a methodology for information retrieval research compiled from 20 years of history. He covered central concepts such as the existence of a system, documents and requests, and the measurement of performance based on relevance judgments assigned by human evaluators. Traditionally, the area of information retrieval has been assessed through empirical demonstrations based on the comparison of models over test collections [240]. Evaluation forums like TREC or INEX have, over time, provided several tracks, with their own test collections, focusing on particular challenges of information retrieval, such as entity retrieval or question answering (see Section 2.3.3.1). Each collection usually contains a set of documents and/or entities to be indexed, a set of topics expressing information needs, and a set of relevance judgments for each topic and associated documents selected from the collection. Topis usually provide a short title, a description, and sometimes also a narrative. Other entity-oriented search tasks, like entity ranking (e.g., from TREC 2018 News track), or list completion (e.g., from INEX 2009 XML Entity Ranking track) might also require topics containing sets of entities or target entity types. Relevance judgments, associated with topics, are usually expressed as

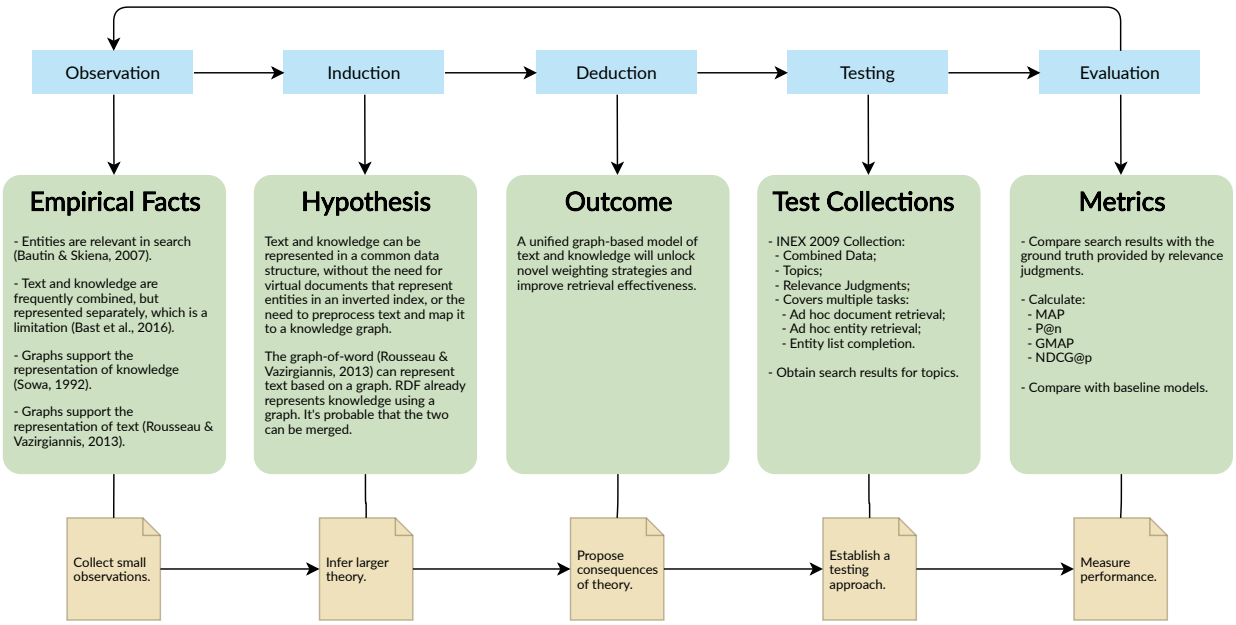



**Figure 3.1:** Empirical cycle applied to graph-based entity-oriented search.

grades between zero and three, assigned by human evaluators to each document, based on its relevance to the given query.

Test collections are a fundamental part of the empirical cycle and, in particular for this thesis, the requirements are demanding. On one side, we need to consider collections with documents, as well as information about entities (i.e., we require combined data). On the other side, in order to explore the general representation aspect provided by graph-based models, we also require relevance judgments for multiple retrieval tasks over the same test collection. Figure 3.1 instantiates the empirical cycle for the problem of improving retrieval effectiveness within entity-oriented search tasks, over graph-based retrieval models. The cycle includes the collection of small observations (empirical facts), an induction process where a larger theory is inferred (an hypothesis is formed), a deduction process where particular consequences of the theory are proposed, a testing stage (the experimentation strategy), and an evaluation stage (the interpretation of statistical evidence gathered during the testing stage).

### 3.1.1 Observation, induction and deduction

The main motivation for this doctoral work had its origin in a simple observation, which is described in Figure 3.1 as the first step of the empirical cycle. While doing prior research in entity-oriented search, we had noticed the disconnect between unstructured data from corpora and structured data from knowledge bases. Through induction we then formulated the hypothesis that a graph could solve the heterogeneity problem of representing documents and entities, and their relations, thus providing a joint indexing strategy that would be beneficial for harnessing all available information and providing a means for cross-referencing information independently of their representation. We deduced that, through such a graph-based model, we would be able to unlock new possibilities for cross-referencing information, providing novel ranking strategies that could improve retrieval effectiveness.

During the course of this work, and following the first iterations of the empirical cycle, we observed that the graph-based solution we proposed would scale poorly, regarding the growth in number of edges in relation to the number of nodes. Accordingly, we proposed a new hypothesis, that a hypergraph could be a solution to

**Table 3.1:** Baselines for evaluating entity-oriented search tasks.

| Task | Index | Ranking |
|---|---|---|
| Ad hoc document retrieval | Lucene index with *doc_id*, *title* and *text* from original document. | TF-IDF / $BM25_{k_1=1.2,b=0.75}$ |
| Ad hoc entity retrieval | Lucene index with *uri*, *label* and *text*, where *text* contains an entity profile built from all sentences in the collection containing the entity *label*. Building the entity profile depends on the index used for ad hoc document retrieval, for retrieving documents containing the entity *label*. | TF-IDF / $BM25_{k_1=1.2,b=0.75}$ |
| Entity list completion | Same index used for ad hoc entity retrieval. | Query: concatenated entity profiles for all input entities, identified by their labels.<br>TF-IDF / $BM25_{k_1=1.2,b=0.75}$ |

reduce the number of edges, by grouping similar nodes and largely reducing the number of (hyper)edges — instead of limiting our modeling approach to binary relations, we focused on expressing n-ary relations instead (e.g., synonyms). We then repeated the testing and evaluation stages and continued relying on the empirical cycle throughout this work.

### 3.1.2 Testing and evaluation

In order to support the research and development, as well as the testing and evaluation of innovative solutions for entity-oriented search (graph-based or not), we have built a workbench called Army ANT, that we describe in Section 5.2. This system is provided as an open source solution and as central code repository for this doctoral work. It supports the full cycle of experimentation, including test collection reading, indexing, searching, learning about the ranking function, and evaluating the system by launching runs for multiple parameter configurations. Initial steps included the implementation of an adapted version of graph-of-word model by Rousseau and Vazirgiannis [16] along with the graph-of-entity model, that we propose as an alternative for combined data (Chapter 6).

While test collections can be used for offline evaluation, there are also options for online evaluation. In particular, there is A/B testing [266], where different groups of users are assigned to two different search engine versions, the original and the experimental version. This enables the measurement of the impact of introduced innovations. As an alternative, there is also team-draft interleaving [245], which combines results from the original search engine with results from an experimental search engine, measuring the outcome as the ratio between clicks in results provided by the experimental version (wins) and the total number of clicks (wins+losses). The first approach (A/B testing) enables an assessment of any search engine component, including the user interface, while the second approach only measures the difference in quality of the ranking function. In this doctoral work, we mostly focus on offline evaluation, but we also take advantage of team-draft interleaving.

**BASELINES** Our tests involved issuing queries built from the provided topics, usually relying on the title of the topic, or the entities or target entity types, depending on the task. In order to position our proposed graph-based models in regard to the state of the art, for each change or tweak to the system, we always computed one or more of the baselines described in Table 3.1, according to the task or tasks being tested. Overall, we rely on simple baselines supported on Lucene indexes, using TF-IDF and BM25 as the ranking functions.

**DATASETS** During the initial development stage, we relied on a small collection without qrels, that we used only for the manual testing of Army ANT and the initial implementations of retrieval models. Wikipedia Relation Extraction Data v1.0[1] was created by Culotta et al. [267] to experiment with probabilistic relation extraction. As such, it provides passages for Wikipedia articles, annotating entity mentions with their official Wikipedia page name and a relation between the entity represented by the source article and the target annotated entity. As combined data, this dataset provided text, entities and their relations. As such, it could be used to create a graph, through the graph-based text and knowledge representations, already identified at the observation stage. We were able to build Army ANT around this data, however, since no relevance judgments were available for entity-oriented search tasks, we switched to INEX 2009 Wikipedia collection[2] [112] as our main test collection, which provided all the required relevance judgments for the different tasks that required evaluation. In Chapter 4, datasets are described in further detail, covering the subsets that we prepared, as well as the associated sampling strategies.

**TREC RUNS** We also took advantage of the TREC evaluation forum to experiment with our retrieval models, both taking advantage of offline and online evaluation. In particular, we participated in TREC 2017 OpenSearch track [268], where team-draft interleaving was used to compare graph-of-word [16] and graph-of-entity [269]. We also participated in TREC 2018 Common Core track [270], where offline evaluation based on the new TREC Washington Post Corpus[3] was used to compare a text-only version of the hypergraph-of-entity with a version that also contained entities and relations. Further details about the TREC participations can be found in Chapters 6 and 9. The test collection used in TREC 2018 is described in Chapter 4.

**METRICS** Our goal was to optimize effectiveness, while monitoring efficiency. We did this by measuring the Mean Average Precision (MAP) and the Precision at a cutoff of $n$ (P@$n$), which we used as the main effectiveness indicators. Both metrics have a comprehensive definition and they provide the required granularity for a retrieval model that is still in the design stage, as opposed to the fine-tuning stage. Additionally, we included the Geometric Mean Average Precision (GMAP), which we used as an indicator of outliers driving MAP up or down, given the lower sensitivity of the geometric mean to outliers, as well as arguments against comparing relative improvements of arithmetic means [271, §2.4]. Despite arguments against using average precision, specifically regarding the fact that it is not an interval scale [271], we decided to focus on this metric, not only because of its comprehensive definition, but also because it provides results that are comparable to past participations in evaluation forums, or to work relying on the same public test sets. We also included the Normalized Discounted Cumulative Gain at a cutoff of $p$ (NDCG@$p$) in order to account for non-binary relevance grades, which some relevance judgment files supply (e.g., for document retrieval, INEX 2010 Ad Hoc track relevance judgments only provide binary relevance grades of 0 or 1, while, for entity ranking and for list completion, INEX 2009 XER track provides relevance grades of 0, 1 or 2). Table 3.2 describes the effectiveness metrics that we mentioned, providing their aggregated formulas, using a normalized notation, for a set of topics with relevance judgments.

In order to monitor efficiency, we simply measured indexing and search time. In particular, we collected the average indexing time per document, as well as the total indexing time for the collection. We also collected the average query time and the total query time for a set of topics. Additionally, and specifically in the context of graph-based models, we sometimes also used the relation between the number of nodes and the number of edges as an indicator of efficiency. For example, when

1 http://cs.iit.edu/~culotta/data/wikipedia.html
2 http://inex.mmci.uni-saarland.de/data/documentcollection.html
3 https://trec.nist.gov/data/wapost/

**Table 3.2:** Retrieval effectiveness metrics.

| Metric | Description | Formula |
|---|---|---|
| MAP [272] | For each query $q \in Q$, with $\|L\|$ returned results, compute P@n for $n = (1, 2, \ldots, \|L\|-1, \|L\|)$ and then calculate the average precision by summing the P@n values and dividing by the total number of relevant documents in the collection. Repeat this for all queries and then aggregate using the mean. | $MAP = \frac{1}{\|Q\|} \sum_{q=1}^{\|Q\|} AP_q$ |
| P@n | Relative precision that accounts for the fraction of R@n relevant documents within the top n results. | $P@n = \frac{1}{\|Q\|} \sum_{q=1}^{\|Q\|} \frac{\|R_q@n\|}{n}$ |
| GMAP [273] | Same as MAP, but the geometric mean is instead used to aggregate average precisions. Average precisions are still calculated using the arithmetic mean. | $GMAP = \left( \prod_{q=1}^{\|Q\|} AP_q \right)^{\frac{1}{\|Q\|}}$ |
| NDCG@p [274] | DCG@p is computed for a query based on the sum of the relevance grades affected by an exponential decay factor from rank 1 to rank p. Normalization is then achieved by ordering the relevance grades in descending order and recomputing the IDCG@p for that ideal ranking (hence the 'I'), which becomes the denominator in NDCG@p. The NDCG@p can then be averaged over all queries. | $DCG@p = \sum_{i=1}^{p} \frac{2^{r_i} - 1}{\log_2(i+1)}$<br>$IDCG@p = DCG@p, r_i \in desc(R)$<br>$NDCG@p = \frac{1}{\|Q\|} \sum_{q=1}^{\|Q\|} \frac{DCG_q@p}{IDCG_q@p}$ |

considering the same collection, if we created a graph with 10 times more edges than nodes and another one with 5 times more edges than nodes, we considered the latter model to have the potential to be more efficient due to lower complexity.

## 3.2 SYSTEMATIC DOCUMENTATION

We relied on DokuWiki[1] for documenting this doctoral work. We systematically organized information about literature, collections, and experiments, providing templates for reading sheets, collection descriptions, and experiment note-taking and result archival. We thoroughly exploited links and backlinks, establishing relations between authors or conferences and their publications, or collections and their mentions in other pages. We also linked subsets to their original dataset, or related experiments among themselves.

In Section 3.2.1, we cover the literature review methodology, describing the reading sheet creation process. In Section 3.2.2, we cover a lightweight approach to documenting collections based on a description sheet template, which includes information about subsets and evaluation results taken from the literature. Finally, in Section 3.2.3, we describe a note-taking and archival strategy for experiments.

### 3.2.1 Literature review for information retrieval

We relied on an exploratory literature review approach, focusing and refining along the process, as concepts became clearer. We used academic search engines to issue queries in an attempt to solve our information needs about entity-oriented search or network science approaches that could be useful for graph-based models. Resulting publications were selected by reading the title, the abstract, the conclusions, and sometimes a part of the introduction, in this order. Selected publications were then added to the DokuWiki, along with a list of specific goals in the form of to-do tasks, to be reviewed in order of priority regarding ongoing research work, or based on the overall relevance and informational value to the thesis. A reading sheet was created for each publication, containing a standardized table of information, as well as reading notes organized according to the structure of the sections of the publication. We frequently included block quotes highlighting important information, and we

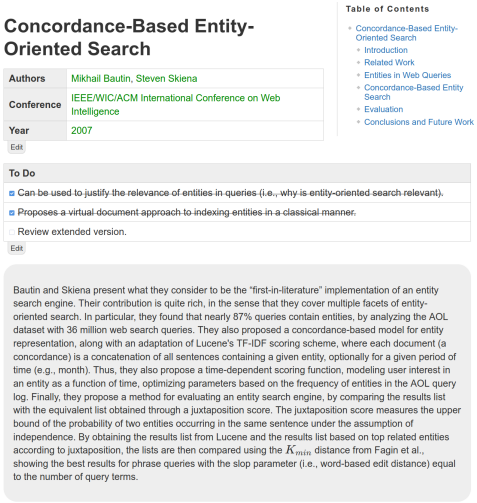
1 https://www.dokuwiki.org/

always finished with a summary paragraph of the work. Next, we describe this strategy in further detail, step by step.

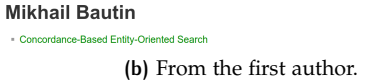
1. **Search Google Scholar**[1] and Semantic Scholar[2] based on queries like:
   - `[ entity retrieval ]`
   - `[ graph-based information retrieval ]`
   - `[ entity-based ranking ]`
   - `[ semantic search ]`
   - `[ entity-oriented search ]`
2. **Follow Google Scholar alerts** for queries like:
   - `[ “Entity-Oriented Search” OR “Entity Ranking” OR “Entity Search” OR “Entity Retrieval” ]`
   - `[ intitle:“Semantic Search” ]`
   - `[ intitle:PageRank ]`
   - `[ text entity graph hypergraph ]`
   - `[ “hypergraph embedding” ]`

   And **Semantic Scholar alerts** for the “Information Retrieval” topic.
3. **Do a quality assessment**, selecting publications based not only on the title, abstract and conclusions, but also on whether they are indexed by well-known bibliographic databases:
   a) DBLP
   b) ACM Digital Library
   c) IEEE Xplore Digital Library
   d) Scopus
   e) Web of Science
4. **Review priority publications** on the broad subjects of the thesis, including relevant surveys, and conference or journal articles, that cover relevant or similar approaches (i.e., graph-based approaches to search, or the combination of corpora and knowledge bases).
   a) Each reviewed publication is added to a DokuWiki, and a reading sheet is created in a page named with the lowercase, dash-separated title of the publication, discarding non-alphanumeric characters.
   b) Each reading sheet contains a table with links to author pages, a bibliographic collection page (i.e., journal, conference, institution) and a year page, following the same naming convention as described above.
   c) Each author, collection and year page contain a list of all backlinks to that page (i.e., all the entries by an author, in a collection, or published in the given year). Optionally, these pages can also include relevant information about the author, the collection or even the year (e.g., a summary of the discoveries or state of the art in that year).
   d) Below the reading sheet table, we replicate the publication’s sections and take notes on each section during reading.
   e) After reading a publication, we insert a written summary of the whole publication below the reading sheet table (and above the notes).

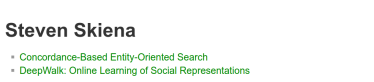
1 https://scholar.google.pt/
2 https://www.semanticscholar.org/



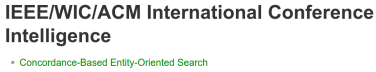

**(a)** Reading sheet.



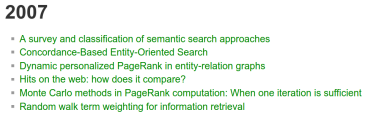

**(b)** From the first author.




**(c)** From the second author.




**(d)** From the same conference.




**(e)** From the same year.

**Figure 3.2:** Systematic documentation of reviewed literature.

f) Links to reviewed publications are displayed in bold, while links to ongoing or short-term reviews are prefixed with a bold label for either **[In Review]** or **[To Review]**.

Figure 3.2 illustrates the type of content created during the review process.

5. **Group selected publications** into relevant subjects and briefly summarize all publications, delving into more detail as deemed relevant.
   a) Compile written summaries into a logical sequence.
   b) Contextualize each publication in regard to the thesis.

**WIKI CHANGES FOR PHD:BIBLIOGRAPHY** Given the nature of a wiki, we have access to the changelog, which we analyzed for the *phd:bibliography* namespace, in order to better understand the process of note taking during literature review. As we can see in Figure 3.3, most of the activity is condensed around the beginning of the doctoral work, around 2016/2017, progressively decreasing for 2017/2018, where a more experimental stage began, and slightly increasing again at the beginning of 2018/2019, carrying light activity up to 2020, the date of writing of this thesis. Overall, this is what is expected from the research process. In the beginning, we had less information about the topic of research, thus we were required to research more. Then, there was enough information for experiments, leading to a new, less intensive cycle of research and another stage of experiments. Nearing the end of the work, we still follow the literature of the area closely, sometimes complementing with additional publications or further researching historical topics within the domain.

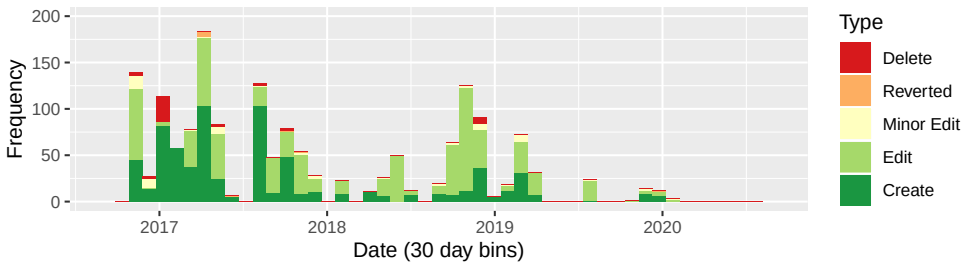
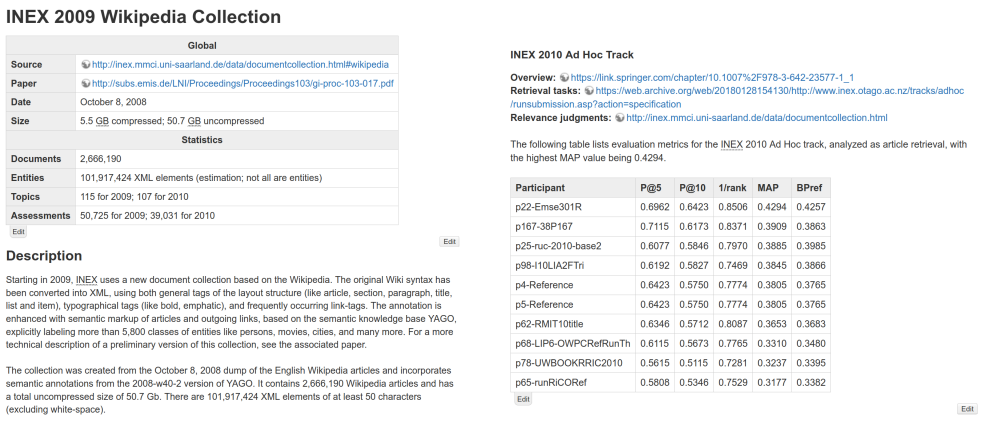



**Figure 3.3:** Doctoral wiki changes over time for the *phd:bibliography* namespace.



**INEX 2009 Wikipedia Collection**

| Global | |
|---|---|
| Source | http://inex.mmci.uni-saarland.de/data/documentcollection.html#wikipedia |
| Paper | http://subs.emis.de/LNI/Proceedings/Proceedings103/gi-proc-103-017.pdf |
| Date | October 8, 2008 |
| Size | 5.5 GB compressed; 50.7 GB uncompressed |
| **Statistics** | |
| Documents | 2,666,190 |
| Entities | 101,917,424 XML elements (estimation; not all are entities) |
| Topics | 115 for 2009; 107 for 2010 |
| Assessments | 50,725 for 2009; 39,031 for 2010 |

Edit

**Description**

Starting in 2009, INEX uses a new document collection based on the Wikipedia. The original Wiki syntax has been converted into XML, using both general tags of the layout structure (like article, section, paragraph, title, list and item), typographical tags (like bold, emphatic), and frequently occurring link-tags. The annotation is enhanced with semantic markup of articles and outgoing links, based on the semantic knowledge base YAGO, explicitly labeling more than 5,800 classes of entities like persons, movies, cities, and many more. For a more technical description of a preliminary version of this collection, see the associated paper.

The collection was created from the October 8, 2008 dump of the English Wikipedia articles and incorporates semantic annotations from the 2008-w40-2 version of YAGO. It contains 2,666,190 Wikipedia articles and has a total uncompressed size of 50.7 Gb. There are 101,917,424 XML elements of at least 50 characters (excluding white-space).

Edit

**INEX 2010 Ad Hoc Track**

**Overview:** https://link.springer.com/chapter/10.1007%2F978-3-642-23577-1_1
**Retrieval tasks:** https://web.archive.org/web/20180128154130/http://www.inex.otago.ac.nz/tracks/adhoc/runsubmission.asp?action=specification
**Relevance judgments:** http://inex.mmci.uni-saarland.de/data/documentcollection.html

The following table lists evaluation metrics for the INEX 2010 Ad Hoc track, analyzed as article retrieval, with the highest MAP value being 0.4294.

| Participant | P@5 | P@10 | 1/rank | MAP | BPref |
|---|---|---|---|---|---|
| p22-Emse301R | 0.6962 | 0.6423 | 0.8506 | 0.4294 | 0.4257 |
| p167-38P167 | 0.7115 | 0.6173 | 0.8371 | 0.3909 | 0.3863 |
| p25-ruc-2010-base2 | 0.6077 | 0.5846 | 0.7970 | 0.3885 | 0.3985 |
| p98-I10LIA2FTri | 0.6192 | 0.5827 | 0.7469 | 0.3845 | 0.3866 |
| p4-Reference | 0.6423 | 0.5750 | 0.7774 | 0.3805 | 0.3765 |
| p5-Reference | 0.6423 | 0.5750 | 0.7774 | 0.3805 | 0.3765 |
| p62-RMIT10title | 0.6346 | 0.5712 | 0.8087 | 0.3653 | 0.3683 |
| p68-LIP6-OWPCRefRunTh | 0.6115 | 0.5673 | 0.7765 | 0.3310 | 0.3480 |
| p78-UWBOOKRRIC2010 | 0.5615 | 0.5115 | 0.7281 | 0.3237 | 0.3395 |
| p65-runRICORef | 0.5808 | 0.5346 | 0.7529 | 0.3177 | 0.3382 |

Edit

Edit

**(a)** Main description. **(b)** Evaluation from the literature.

**Figure 3.4:** Systematic documentation of the INEX 2009 Wikipedia collection.

### 3.2.2 Collections: description sheets, subsets and evaluations

We created a page in the collections section of the wiki for each dataset that we used, created, or otherwise explored. In order to ensure consistency in the description, we prepared a template containing a table of metadata to be filled about each collection, along with a longer textual description. The fields that we considered were the following: *Source*, *Paper*, *Date*, and *Size*. We also added fields for statistics specific to test collections within the domain of entity-oriented search: *Documents*, *Entities*, *Topics*, and *Assessments*. For other types of data, these statistics were ignored or replaced. For example, for network data, we used *Nodes* and *Edges* instead. Figure 3.4a illustrates the template that we have just described, in its application to the INEX 2009 Wikipedia collection.

Due to scalability issues, we also relied on multiple subsets of our main test collection. Each subset was described in the wiki as any other collection, assigning to the *Source* descriptor a link to the page of the original dataset, instead of a URL to the source website or similar resource. Sampling strategies are further described in Chapter 4, along with a list of all the generated subsets that we used.

Whenever they were available, we also included evaluation metrics found within the literature, if applied to one of the entity-oriented search tasks that we listed in Section 1.3.2, for the same dataset. This information is usually available in the overview papers [68–70, 142, 247, 248, 275, 276] from the evaluation forums that provided the test collection. Figure 3.4b shows the effectiveness evaluation metrics for the ad hoc document retrieval task in INEX 2010 Ad Hoc track [248].

**WIKI CHANGES FOR PHD:COLLECTIONS** We described seven datasets in the wiki, categorizing them as COMBINED DATA (four datasets), NETWORKS (two datasets), and QUERY LOGS (one dataset). We also prepared five subsets of our main test collection, INEX 2009 Wikipedia collection, which we describe in Section 4.1.1. Dataset information was organized over time in the wiki as shown in Figure 3.5[1]. As we can see, the effort of providing basic dataset descriptions was low. At the beginning of 2017/2018, we introduced the first significative amount of metadata, with minor updates throughout the academic year. At the beginning of the 2018/2019 and 2019/2010 academic years, there were noticeable updates to the *phd:collections* namespace, probably corresponding to evaluation results based on the documented test collections, that we only added at a more final stage.

1 In order to make the volume of changes over time comparable, we used the same scale as Figure 3.3 for all the doctoral wiki plots.

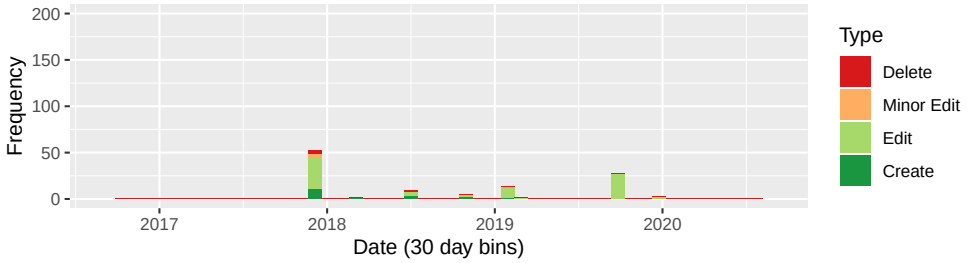



**Figure 3.5:** Doctoral wiki changes over time for the *phd:collections* namespace.

### 3.2.3 Experiments: taking notes and archiving results

Following a similar philosophy to the literature review and collection description, we also established a template for the documentation of experiments in the wiki. Each experiment page contains a metadata table (Figure 3.6a) with the following fields: *ID* (e.g., "Experiment 1"), *Start date* (e.g., "2017-10-24 16:38"), *End date* (e.g. "Ongoing"), *Why do it?* (motivation; usually in the sequence of a previous experiment), *Main strengths* (expected improvements), *Main weaknesses* (expected issues), and *Test collection* (a link to a wiki collection page). One weakness of this approach is the lack of structure for follow-up or sub-experiments, which we suboptimally documented through research logs.

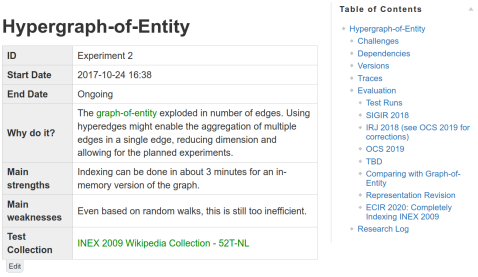
Each experiment also included a to-do list, a description table of the model versions explored (Figure 3.6d), and an evaluation section with performance metrics for each of those versions. We also added tables to describe predicted or identified challenges, dependencies that would block the execution of the experiment, and trace logs from Army ANT's ranking function learn mode, which are be described in Section 5.2. Over time, the challenges and dependencies sections were mostly deprecated in favor of the to-do list, and the archival of trace logs proved cumbersome and with little utility, thus being increasingly ignored over time.

On the other side, the archival of results from Army ANT's evaluation tasks (Figure 3.6b) was particularly useful, for instance for verifying the calculation of effectiveness metrics through `trec_eval`[1]. We stored a metrics summary file in CSV and TeX formats, representing a table of runs and effectiveness metrics. When downloading these table files from Army ANT, we used 10 decimal places and included all available metrics and additional columns. We also stored a ZIP file per run, containing additional detailed information, such as average precisions used in the calculation of MAP, or efficiency statistics, such as average query time. We also stored the output of index inspections, for instance covering information about the number of paths established between documents for a given hyperedge type (e.g., *synonyms* and *context*).

Finally, we created a section for research logs (Figure 3.6c), where we added links to wiki pages under the hierarchy of the current experiment. Each research log entry represented a reflection on parts of studied models, possibly branching into follow-up or sub-experiments.

**WIKI CHANGES FOR PHD:EXPERIMENTS** Figure 3.7 depicts the experimental activity over time. Most of the experimental activity, according to the doctoral wiki's changelog for the *phd:experiments* namespace, occurred during the academic years of 2017/2018 and 2018/2019, with a few activity at the beginning of 2019/2020. We can see that experimental activity peaked at the beginning of 2017/2018, consistently decreasing, while interpolating with lower activity periods, and it also peaked at the beginning of 2019, until it almost completely stopped. We also compare the experimental activity with the literature reviewing activity (Figure 3.3). We

1 https://github.com/usnistgov/trec_eval



## Hypergraph-of-Entity

| | |
|---|---|
| **ID** | Experiment 2 |
| **Start Date** | 2017-10-24 16:38 |
| **End Date** | Ongoing |
| **Why do it?** | The graph-of-entity exploded in number of edges. Using hyperedges might enable the aggregation of multiple edges in a single edge, reducing dimension and allowing for the planned experiments. |
| **Main strengths** | Indexing can be done in about 3 minutes for an in-memory version of the graph. |
| **Main weaknesses** | Even based on random walks, this is still too inefficient. |
| **Test Collection** | INEX 2009 Wikipedia Collection - 52T-NL |

Edit

Table of Contents

- Hypergraph-of-Entity
  - Challenges
  - Dependencies
  - Versions
  - Traces
  - Evaluation
    - Test Runs
    - SIGIR 2018
    - IRJ 2018 (see OCS 2019 for corrections)
    - OCS 2019
    - TBD
    - Comparing with Graph-of-Entity
    - Representation Revision
    - ECIR 2020: Completely Indexing INEX 2009
  - Research Log

**To Do**

- ~~Characterize an instance of the hypergraph-of-entity.~~
- ~~Measure the stability of results depending on WALK_LENGTH and WALK_REPEATS (compare ranking for several iterations with the same configuration).~~
- ~~Study WALK_REPEATS through Kendall's coefficient of concordance.~~
- ~~Implement feature extraction and streamline integration into the models.~~
- ~~Assign weights to nodes and hyperedges (test several functions until we find a basic one with a good discriminative power).~~
- Characterize weight distributions in order to define thresholds to prune (e.g., bottom 10%).
- ~~Implement biased random walks, as an option.~~
- ~~Implement pruning methods with configurable pruning threshold.~~

**(a)** Main description.

## A General Model for Entity-Oriented Search

Edit

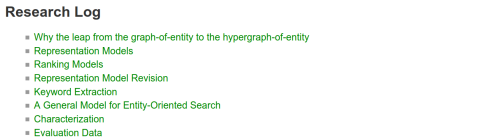
Table of Contents

- A General Model for Entity-Oriented Search
  - Evaluation Data
  - Results

### Evaluation Data

- metrics-20191011t160826.csv
- metrics-20191011t160830.tex
- 5d9346413ee2723623583f9d-lucene_tf_idf-inex_2009-ad_hoc_document_retrieval.zip
- 5d9347113ee2723623583f9e-lucene_bm25-inex_2009-ad_hoc_document_retrieval.zip
- 5da07a833ee2721037ec4441-lucene_tf_idf-inex_2009-ad_hoc_entity_retrieval.zip
- 5da07b573ee2721037ec4443-lucene_bm25-inex_2009-ad_hoc_entity_retrieval.zip
- 5da07bbb3ee2721037ec4444-lucene_tf_idf-inex_2009-entity_list_completion.zip
- 5da07ca63ee2721037ec4445-lucene_bm25-inex_2009-entity_list_completion.zip
- 5d8e25113ee2720157ee6ab1-hgoe_rws-inex_2009-ad_hoc_document_retrieval.zip
- 5d91f8de3ee2723623583f9b-hgoe_rws-inex_2009-ad_hoc_entity_retrieval.zip
- 5d92f4fb3ee2723623583f9c-hgoe_rws-inex_2009-entity_list_completion.zip

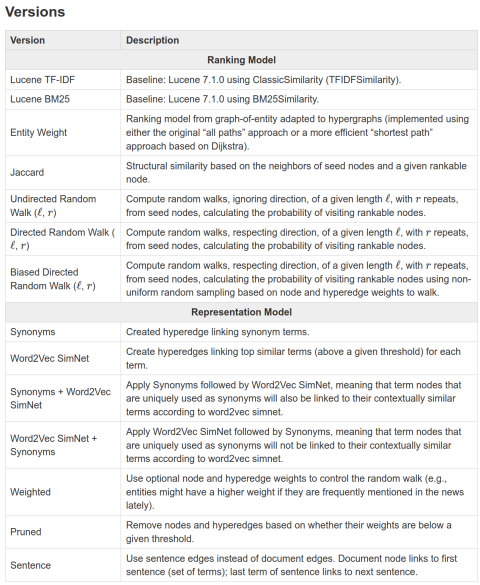
**(b)** Evaluation results archive.



## Research Log

- Why the leap from the graph-of-entity to the hypergraph-of-entity
- Representation Models
- Ranking Models
- Representation Model Revision
- Keyword Extraction
- A General Model for Entity-Oriented Search
- Characterization
- Evaluation Data

**(c)** Research log entries.



## Versions

| Version | Description |
|---|---|
| **Ranking Model** | |
| Lucene TF-IDF | Baseline: Lucene 7.1.0 using ClassicSimilarity (TFIDFSimilarity). |
| Lucene BM25 | Baseline: Lucene 7.1.0 using BM25Similarity. |
| Entity Weight | Ranking model from graph-of-entity adapted to hypergraphs (implemented using either the original "all paths" approach or a more efficient "shortest path" approach based on Dijkstra). |
| Jaccard | Structural similarity based on the neighbors of seed nodes and a given rankable node. |
| Undirected Random Walk ($\ell$, $r$) | Compute random walks, ignoring direction, of a given length $\ell$, with $r$ repeats, from seed nodes, calculating the probability of visiting rankable nodes. |
| Directed Random Walk ( $\ell$, $r$) | Compute random walks, respecting direction, of a given length $\ell$, with $r$ repeats, from seed nodes, calculating the probability of visiting rankable nodes. |
| Biased Directed Random Walk ($\ell$, $r$) | Compute random walks, respecting direction, of a given length $\ell$, with $r$ repeats, from seed nodes, calculating the probability of visiting rankable nodes using non-uniform random sampling based on node and hyperedge weights to walk. |
| **Representation Model** | |
| Synonyms | Created hyperedge linking synonym terms. |
| Word2Vec SimNet | Create hyperedges linking top similar terms (above a given threshold) for each term. |
| Synonyms + Word2Vec SimNet | Apply Synonyms followed by Word2Vec SimNet, meaning that term nodes that are uniquely used as synonyms will also be linked to their contextually similar terms according to word2vec simnet. |
| Word2Vec SimNet + Synonyms | Apply Word2Vec SimNet followed by Synonyms, meaning that term nodes that are uniquely used as synonyms will not be linked to their contextually similar terms according to word2vec simnet. |
| Weighted | Use optional node and hyperedge weights to control the random walk (e.g., entities might have a higher weight if they are frequently mentioned in the news lately). |
| Pruned | Remove nodes and hyperedges based on whether their weights are below a given threshold. |
| Sentence | Use sentence edges instead of document edges. Document node links to first sentence (set of terms); last term of sentence links to next sentence. |

**(d)** Model versions considered for the experiment.

**Figure 3.6:** Systematic documentation of the hypergraph-of-entity experiments (detailed information about these experiments can be found on Chapters 7, 8 and 9).

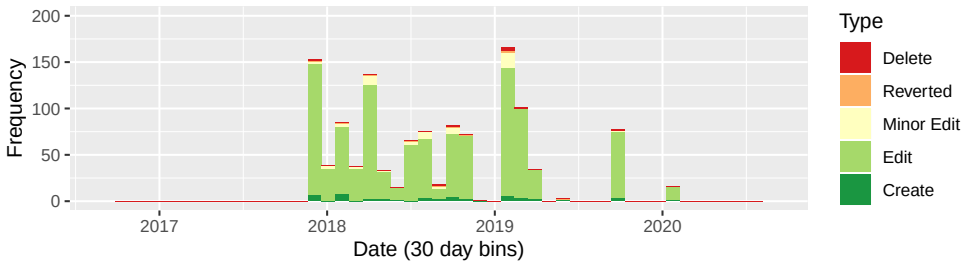



**Figure 3.7:** Doctoral wiki changes over time for the *phd:experiments* namespace.

find that both activity depictions fit in the continuity of time, with literature review leading to experimental activity and complementing it, as expected.

## SUMMARY

In this chapter, we described the research methodology applied in this thesis. We covered the empirical cycle based on test collections, that is so often employed for offline information retrieval evaluation. We also covered online evaluation approaches, where we included A/B testing, as well as team-draft interleaving, out of which only the latter was explored in this thesis. We presented details on the baselines for the different entity-oriented search tasks, we briefly introduced the specific datasets that we relied on, as well as the TREC runs that we participated in, and we listed and briefly described the effectiveness metrics that we used to evaluate the tested models.

Another central element of our research methodology was the systematic documentation approach supported by a DokuWiki instance, that we prepared for this purpose. In particular, we described the literature review methodology, including the reading sheets that we created and added to the doctoral wiki, applied to a limited selection of publications that required a deeper reading or benefited from a more organized reviewing approach. We also described the structure we formulated to store basic information about data collections, as well as the documentation workflow applied to our experiments, which included a research log, an archive of evaluation results, and information about the different model versions that we tested. Finally, in each of the corresponding sections, we included statistics about the changes made to the doctoral wiki over time, illustrating the flow of execution of the detailed methodology.

# 4 DATASETS

## Contents



Information retrieval evaluation is traditionally based on test collections, many of them prepared and provided through the collaboration of the community over evaluation forums (see Section 2.3 for more information on evaluation forums and Section 3.1.2 for a better understanding on the testing and evaluation methodology). Data can also be provided directly through web services, usually as a part of an online evaluation endeavor. Moreover, during the research process, there are newly created datasets, either as a byproduct of developed systems and experiments, based on established collaborations, or simply to respond to specific evaluation needs. Finally, there are also secondary datasets, that are used for small-scale testing during software development, or for training auxiliary classifiers, to name a few examples.

The structure of this chapter is organized as follows:

- **Section 4.1** describes the test collections used to evaluate the performance of retrieval models, either through a set of topics and relevance judgments provided *a priori*, or through a set of topics used to generate results to be evaluated *a posteriori*, online and through team-draft interleaving. It covers INEX 2009 Wikipedia collection [§4.1.1], TREC Washington Post Corpus [§4.1.2], and the data from the Social Science Open Access Repository used in TREC 2017 OpenSearch track [§4.1.3].
- **Section 4.2** covers contributed datasets prepared throughout this thesis, and specifically the Simple English Wikipedia Link Graph with Clickstream Transitions 2018-12 [§4.2.1], which we prepared to evaluate link analysis metrics, like Fatigued PageRank that we propose in this thesis.
- **Section 4.3** presents secondary or auxiliary datasets used for diverse tasks. It covers Wikipedia Relation Extraction Data v1.0 [§4.3.1], which is a small dataset that we used to support the development of Army ANT and the initial retrieval models. It also covers the BBC Dataset [§4.3.2], which was used to train a query classification model for supporting the query log analysis task based on the two previous datasets.

## 4.1 TEST COLLECTIONS

In this section, we describe the test collections used for the assessment of the graph-based models that we introduce in this thesis (Chapters 6 and 7), against the proposed baselines (Section 3.1.2). Each of the test collections that we considered contains unstructured data in the form of text, as well as structured data representing entity mentions and/or relations. This type of data is fundamental for the evaluation of entity-oriented search systems, in particular to support the evaluation of the four tasks that we previously identified (Section 1.3.2). A test collection also contains a set of topics describing information needs, that can be used to issue keyword or entity queries, depending on the task. Relevance judgments must be provided for each of the assessed tasks, which represents an effort for human contributors. One of the challenges in evaluating a general model for entity-oriented search is the availability of assessment data for different tasks over a common collection. This is why some of the test collections that we use are over ten years old. In the following sections, we briefly describe each of the test collections that we considered, in order of relevance to this work, from the most to the least relevant.

### 4.1.1 INEX 2009 Wikipedia collection

The INEX 2009 Wikipedia collection [112] is a snapshot of the English Wikipedia, from October 8, 2008. It contains 2,666,190 articles in XML format with 101,917,424 XML elements. Documents were annotated with semantic information from the 2008-w40-2 version of the YAGO ontology, assigning labels based on one or multiple of the 5,800 available classes (e.g., PERSON, MOVIE, CITY). This resulted in a dataset that requires 50.7 GiB of space when uncompressed, and 5.5 GiB when compressed in four gzipped tar archives of 1.4 GiB each.

Despite being a 10-year-old test collection, INEX 2009 Wikipedia collection is still one of the few datasets that simultaneously provides relevance judgments for ad hoc document retrieval, ad hoc entity retrieval, and entity list completion. It is also adequate for experimenting with a combination of unstructured and structured data, since it contains both text, usually representing an entity, and links among these entities. Furthermore, while we do not take advantage of it in this work, the semantic annotations are also useful for identifying entity types, supporting for example type queries [3, §2.1].

Knowledge bases like Wikipedia (semi-structured), DBpedia (structured), or Wikidata (structured) are frequently used to augment a corpus. A typical preprocessing pipeline in entity-oriented search is to annotate documents through named entity recognition and disambiguation, linking each entity to its entry in the knowledge base. This can then be used to improve document, or entity retrieval. However, when working with an encyclopedic test collection like INEX 2009 Wikipedia collection, the corpus and the knowledge base are, in practice, the same. On one side, this means that it is harder to improve retrieval effectiveness through external knowledge, which is sparser as it is limited to extending local views of the collection. On the other side, it means that the collection doesn't need to be augmented, but rather preprocessed so that text, entities and their relations can be extracted directly from the structure of the XML document.

For assessment, the INEX Ad Hoc track provides 115 topics from the 2009 occurrence, with 50,725 individual relevance judgments [275], and 107 topics from the 2010 occurrence, with 39,031 individual relevance judgments [248]. Each individual relevance judgment contains the query identifier, the document identifier, the number of relevant characters, the offset of the best entry point (usually the first relevant passage) and offset–length pairs for the relevant passages.

For the INEX 2009 and 2010 Ad Hoc tracks, both the topics and the relevance judgments were produced by participants of the evaluation forum. In 2010, only a fraction of the topics (52 out of 107) had associated relevance judgments. Furthermore,

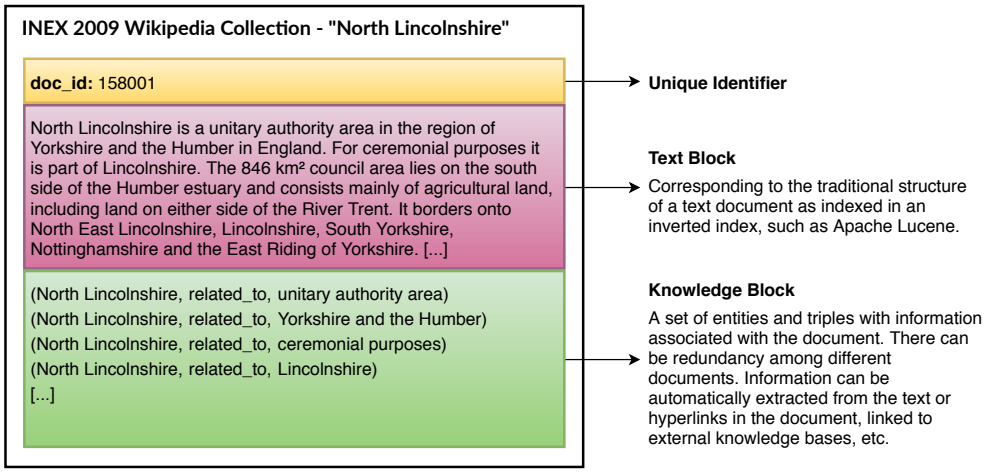



**Figure 4.1:** Extended document definition for combined data. Example from INEX 2009 Wikipedia collection, for the XML representing the Wikipedia article about *North Lincolnshire*.

only seven topics were judged by more than one individual, with the remaining topics being judged by a single individual. Despite the reduced number of judges per topic, it is important to notice that individual passages (not documents) were the main object of judgement. This means that every considered document was explored in detail, which was somewhat transposed to the relevance judgments, since we know exactly which passages led to each judge assigning relevance to the document.

#### 4.1.1.1 *Document and query construction*

In our experiments, we extracted from each XML document its textual content along with links to other documents and three attributes: *doc_id*, *title*, and *url*. XPath was used to extract this information. The *doc_id* was given by `//header/id/text()`, the *title* was given by `//header/title/text()`, and the *url* was built from the entity's Wikipedia page, based on `http://en.wikipedia.org/?curid=`<*doc_id*>. Textual content was extracted based on `//bdy/descendant-or-self::*[not(ancestor-or-self::template)and not(self::caption)]/text()`. Links leading to other documents in the collection were extracted based on `//link/@xlink:href`. This enabled us to map outgoing semantic relations from each document, obtaining a set of triples to accompany the textual content.

Figure 4.1 illustrates the extracted elements from each XML document, forming what we call an extended document for combined data. A regular document usually contains multiple text fields (e.g., *title*, *content*, etc.), which corresponds to the text block in the extended document. However, an extended document also includes a knowledge block, which can contain single entities, as well as triples, that are usually available as structured data in the original document. For the INEX collection, the knowledge block was directly extracted from the XML (we used links to other documents, in order to implicitly build the triples), but in other collections this could be obtained as the result of an information extraction pipeline. There is no restriction about the source of the knowledge block, except that it should represent a set of entities and/or triples related to the document. For example, the triples might represent co-occurring entities in a sentence or paragraph, or statements obtained from a dependency parser, or they could represent external knowledge about identified entities, from an external knowledge base. The knowledge block purely depends on the target application.

For evaluating ad hoc document retrieval, we used the title of each Ad Hoc track topic as a keyword query. This was given by `//topic/title/text()` when applied over the *2010-topics.xml* file. We then assessed effectiveness based on whether or

not retrieved documents contained relevant passages, according to the provided relevance judgments (*inex2010.qrels*). For evaluating ad hoc entity retrieval and entity list completion we relied on the *inex09-xer-topics-final.xml* file from the Entity Ranking track, using a concatenation of the title and the categories, respectively given by `//inex_topic/title/text()` and `//inex_topic/categories/category/text()`, to issue a keyword query for the ad hoc entity retrieval task. For the entity list completion task, we relied on the entities specified under `//inex_topic/entities/entity/text()` to issue an entity query.

#### 4.1.1.2 *Subsets and sampling approach*

Due to our focus on generalization and effectiveness, many of our experimental retrieval models were inefficient and impractical to evaluate over the complete INEX 2009 Wikipedia collection. Thus, we were required to scale down our experiments, relying on smaller subsets of the collection. Our sampling method was based on the topics used for relevance assessment in the 2010 Ad Hoc track. The remaining tasks were evaluated over the complete collection and thus did not require the preparation of subsets. Furthermore, generating separate subsets for each task, while relying on the approach we proposed, would result in different collections being used for evaluating each task, which defeated the purpose of running all experiments over a common representation model (i.e., over a single index), by simply reconfiguring the universal ranking function.

In order to create the subset, we first selected $n$ topics, uniformly at random, from a total of 52 topics with available relevance judgements (out of the 107 topics for 2010). Then, we filtered the relevance judgments, keeping only those regarding the selected topics. Finally, we filtered out documents that were not mentioned in the relevance judgments from each of the four archives (*pages25.tar.bz2*, *pages26.tar.bz2*, *pages27.tar.bz2* and *pages28.tar.bz2*). Our sampling method also provided the option to include all documents linked from the selected documents directly mentioned in the relevance judgments. However, this would significantly increase the size of the subset, defeating the purpose of lowering the scale.

Each subset of the INEX collection was identified by the prefix "INEX 2009" followed by the number of sampled topics (e.g., "10T") and whether or not linked documents were included ("NL" for 'no links' and "WL" for 'with links'). For example "INEX 2009 10T-NL" contains the documents evaluated for 10 random topics, excluding the documents linked to the evaluated documents (unless they were also a part of the same subset of relevance judgments). We prepared the following subsets that we rely on throughout this thesis:

**INEX 2009 10T-NL** Contains 7,487 documents and 7,504 individual relevance judgments for the following 10 topics:

- `[ Monuments of India ]`
- `[ composer museum ]`
- `[ retirement age ]`
- `[ japanese ballerina ]`
- `[ dinosaur ]`
- `[ President of the United States ]`
- `[ European fruit trees ]`
- `[ Einstein Relativity theory ]`
- `[ famous chess endgames ]`
- `[ predictive analysis +logistic +regression model program application ]`

**INEX 2009 52T-NL** Contains 37,788 documents with the whole 39,031 individual relevance judgments for the complete list of 52 topics in the collection. This

**Table 4.1:** TREC Washington Post Corpus content types and their frequency per collection.

| *contents.type* | **News Articles** | **Blog Posts** |
|---|---|---|
| *"author_info"* | 147,189 | 327,422 |
| *"byline"* | 187,322 | 358,337 |
| *"date"* | 236,649 | 358,379 |
| *"image"* | 178,124 | 477,949 |
| *"kicker"* | 236,649 | 357,878 |
| *"sanitized_html"* | 5,462,872 | 6,355,698 |
| *"title"* | 236,649 | 358,232 |
| *"video"* | 31,014 | 120,260 |
| *"gallery"* | 26,266 | 22,613 |
| *"list"* | 349 | 27,928 |
| *"deck"* | 2,065 | 7,378 |
| *"tweet"* | 367 | 170,625 |
| *"instagram"* | 68 | 9,807 |
| *"pull_quote"* | 0 | 231 |
| *"inline_story"* | 0 | 23 |
| *"ar-wikitude"* | 0 | 2 |
| *"top_deck"* | 0 | 5 |
| *"animated_video"* | 0 | 1 |

subset is partially complete, in the sense that it only excludes non-assessed documents, including all topics and relevance judgments.

### 4.1.2 TREC Washington Post Corpus

The TREC Washington Post Corpus[1] was first used during TREC 2018 in the Common Core and News tracks. It contains 608,180 documents collected from the Washington Post between January 2012 and August 2017. Out of these documents, we only used the subset of 595,037 documents (236,649 news articles and 358,388 blog posts), that resulted of the removal of duplicates by *id* that had been a part of the initial release of the collection. Multiple duplicates by *title* still exist in the collection. However, the TREC Common Core track evaluation data available for the Common Core and News tracks was prepared over this subset.

The news article or blog post text is split by paragraphs, which are stored as content items, including HTML tags. Resources like images, video, or embedded media are also identified as separate content items, distinguished through a *type* attributed. In total the collection requires 1.5 GiB of compressed space for storage, or 6.9GB when uncompressed.

The documents are stored as JSON and include the following fields:

- *id* – a unique document identifier;
- *title* – the title of the blog post or news article;
- *author* – author(s), as extracted from the byline of the publication;
- *published_date* – UNIX timestamp corresponding to the date of publication;
- *type* – collection identifier, either *"blog"* or *"article"*;
- *article_url* – hyperlink pointing to the web page of the publication;
- *source* – fixed value, *"The Washington Post"*;
- *contents* – objects containing information for the given *type* (see Table 4.1).

1 https://trec.nist.gov/data/wapost/

As we can see in Table 4.1, the most frequent element is *"sanitized_html"*, which usually corresponds to paragraphs (5,116,047 out of 5,462,872 for news articles and 5,978,278 out of 6,355,698 blog posts). When considering the frequency of content of type *"tweet"* and *"instagram"*, we also verify that social media shares are more frequent in blog posts than in news articles. Only blog posts contain content of type *"pull_quote"*, *"inline_story"*, *"ar-wikitude"*, *"top_deck"*, and *"animated_video"*, despite its overall low frequency.

Topics and relevance judgments were also provided for this collection during the Common Core and News tracks occurring in 2018. For the ad hoc document retrieval task, 50 topics were provided in the context of the Common Core track, comprised of 25 topics from the 2017 occurrence, and 25 new topics prepared by NIST assessors in 2018. Along these topics, 26,233 individual relevance judgments were also provided with grades ranging from 0 to 2, out of which 85% corresponded to non-relevant documents (i.e., with grade 0). The News track also provided topics and relevance judgments for a background linking task and an entity ranking task. The background linking task had the goal of retrieving similar documents that provided useful background information for a given document. For this task, 50 topics in the form of documents represented by their *docid* and *url* were provided, along with 8,508 individual relevance judgments, where the relevance score (rel) ranged between 0 and 4, as usual, however the provided grade in the qrels file was $2^{\mathrm{rel}}$ (i.e., 0, 2, 4, 8 or 16), with non-relevant documents being the exception, as their grade was set to zero. The entity ranking task had the goal of ranking a set of entities previously extract from a document in the collection, according to their usefulness as background information to that document. For this task, 50 topics were provided in the form of documents represented by their *docid* and *url*, accompanied by a list of entities represented by their *id*, *mention* and *link*. For assessment, 82 individual relevance judgments were provided, with entities graded from 0 to 4, from least to most useful in the context of the given document. Since no explicit keyword query was used to rank the entities (besides the document's text), it was not straightforward to adapt this data for the evaluation of ad hoc entity retrieval, although we considered it an option.

#### 4.1.2.1 *Link graph*

Some of our experiments also relied on a link graph from TREC Washington Post Corpus. In order to build the link graph, we translated each hyperlink into the corresponding document *id*, which acted as a node identifier, otherwise ignoring the hyperlink when it did not point to a valid document in the corpus. This resulted in a graph with 159,228 nodes and 319,985 edges, with an average outdegree of $2.01 \pm 2.63$ and an average indegree of $2.01 \pm 7.87$. For news articles, the average outdegree was $1.43 \pm 2.28$ and the average indegree was $2.39 \pm 9.57$, while, for blog posts, the average outdegree was $2.22 \pm 2.71$ and the average indegree was $1.87 \pm 7.15$. Overall, there was a lower number of outgoing links departing from news articles. On the other hand, news articles also received the highest number of incoming links.

### 4.1.3 TREC 2017 OpenSearch SSOAR

The TREC 2017 OpenSearch SSOAR dataset was prepared using the Living Labs API[1] to collect documents, queries, and document lists pre-ranked for those queries, for the Social Science Open Access Repository (SSOAR) site. In the evaluation forum, there were several rounds where participant teams could submit runs with their results for the given set of topics (the queries). Evaluation was based on

1 Living Labs is an open source framework for the evaluation of information retrieval systems based on an interleaving approach where the participant's results are combined with the results provided by the site. Living Labs is available at https://bitbucket.org/living-labs/ll-api.

team-draft interleaving [245], where results for the participant (e.g., FEUP) and site (e.g., SSOAR) were interleaved for new searches over the provided queries. These queries frequently corresponded to the top issued queries in the site, so that they could have a high chance of being reissued. This created evaluation opportunities, without the need for a web service infrastructure provided by participants, which could jeopardize the normal activity of the site, efficiency-wise. This way, each time an interleaving from a participant was shown to a user, it formed an impression. During this impression, the user could: click no results (tie), a higher number of participant results (win) or a higher number of site results (loss). Wins, losses and ties were accounted for each participant, providing a final outcome value, that could be used for evaluation, representing the fraction of participant wins over the total clicks.

In the dataset that we prepared, we also included the detailed feedback, with all of the displayed results, identifying individually clicked results, as well as the aggregated outcome provided for our runs. Our locally organized version of the API data, which only required 28 MiB, was prepared during our participation in the evaluation forum. It resulted in a MongoDB database with the following collections: *docs*, *queries*, *feedback*, *dead_period_feedback*, and *extra_round_feedback*. This included 32,492 documents with publication dates ranging from August 29, 2012, to May 26, 2017, as well as 1,165 queries, out of which 676 were labeled as *"train"* and 489 were labeled as *"test"*. It also included three sets of feedback impressions for the runs that we submitted as team FEUP: 59 for official rounds (18 with clicks); 4,683 for a dead period (29 with clicks); and 97 for an extra round (25 with clicks). The dead period and the extraordinary round were provided by the organization due to a technical issue that we further detail in Section 6.3.2.1. The dead period corresponded to a time between July 17, 2017 and July 31, 2017, where no official rounds were taking place. The extraordinary round corresponded to an official round added to the track due to the technical issues, and it occurred during October 2017, resulting in feedback over a period ranging from July 25, 2017 to November 13, 2017.

#### 4.1.3.1 *Data collection details*

Data was provided to participants through the Living Labs API in JSON format. In particular, an array of documents could be accessed through a `GET` request to `/api/v2/participant/docs`. Each document contained a *site_id* (*"ssoar"* for the 2017 occurrence of the OpenSearch track), a *doc_id*, a *creation_time*, a *title*, and a *content* object. The *content* object contained relevant metadata about the document, including the *abstract* and other fields like *subject* or *type*. We divided all available metadata into a text block (*title* and *abstract*), and a knowledge block (*author*, *language*, *issued*, *publisher*, *type*, *subject* and *description*). This resulted in a total of 223,749 entities, obtained from the fields selected for the knowledge block.

We also retrieved the train and test queries from `/api/v2/participant/query`, generating rankings for each query based on our implementation of the graph-of-word and graph-of-entity (detailed in Chapter 6). Finally, we added any missing documents to the end of the results list, based on the provided rankings for each query, available at `/api/v2/participant/doclist/<qid>`. The *doclist* rankings corresponded to candidate documents, provided by the site, that could be used for instance for reranking and should be a part of the submitted runs (our approach did not, however, take advantage of this information). Run submission was done through a `PUT` request, per query, to `/api/v2/participant/run/<qid>`. Feedback could then be accessed through a `GET` request to `/api/v2/participant/feedback/<qid>`, per query and, optionally, for a specific run. We were also directly offered a dump with data from the dead period, due to the technical issue that we have already mentioned.

## 4.2 CONTRIBUTED DATASETS

In this section, we describe the link graph that we prepared for an in-memory, lower scale evaluation of link analysis approaches like PageRank. This test graph, built from Wikipedia hyperlinks and weighted based on clickstream data, replicated the evaluation approach by Dimitrov et al. [157].

### 4.2.1 Simple English Wikipedia Link Graph with Clickstream Transitions 2018-12

The Simple English Wikipedia Link Graph [277] was built from the *page*[1] and *pagelinks*[2] SQL dumps of the Simple English Wikipedia from January 1st, 2019, using page names as node identifiers and considering only pages within the *article* namespace, with links from other pages. The SQL query used to prepare the graph was the following:

```
SELECT p1.page_title AS SOURCE, pl_title AS target
FROM pagelinks
JOIN page AS p1 ON pl_from = p1.page_id
WHERE pl_namespace = 0 AND pl_from_namespace = 0
```

The graph was then stored as a gzipped GML file. Given no clickstream data was directly available for the Simple English Wikipedia, we used the clickstream data for the English Wikipedia from December 2018[3], adding a *transitions* attribute to each edge, that would be zero whenever information was unavailable. This resulted in a graph with 897,577 nodes and 6,986,460 edges, with an average outdegree of 7.78±49.52 and an average indegree of 8±62, based on information not only from existing pages at the date, but also from deleted pages that had been linked to those active pages. This justifies the direct usage of *pl_title* instead of a *page_id* attribute in table *pagelinks*, since these entries would not have a corresponding entry in the *page* table.

## 4.3 OTHER DATASETS

In this section, we briefly describe secondary or auxiliary datasets that we used throughout this doctoral work. In particular, we describe the following datasets:

- Wikipedia Relation Extraction Data v1.0 [§4.3.1], which we used during the development stage to test the Army ANT evaluation framework and the first implemented retrieval models;
- BBC Dataset [§4.3.2], which we used to train a query classifier based on five news categories.

### 4.3.1 Wikipedia Relation Extraction Data v1.0

Wikipedia Relation Extraction Data v1.0[4] [267] is a dataset created in June 10, 2006 by the University of Massachusetts and Google. It contains 1,110 paragraphs for entities described in 441 Wikipedia articles. The top 5 entities with the most paragraphs are: *George W. Bush*, *Charles Darwin*, *Prescott Bush*, *Dick Cheney*, and *Kurt Cobain*. Each paragraph contains anchors that were annotated with an additional

1 https://dumps.wikimedia.org/simplewiki/20190101/simplewiki-20190101-page.sql.gz
2 https://dumps.wikimedia.org/simplewiki/20190101/simplewiki-20190101-pagelinks.sql.gz
3 https://dumps.wikimedia.org/other/clickstream/2018-12/clickstream-enwiki-2018-12.tsv.gz
4 http://cs.iit.edu/~culotta/data/wikipedia.html

*relation* attribute, establishing the type of dependency between the source entity, from where the paragraph was extracted, and the target entity, pointed to by the anchor. In total, there are 4,681 annotated relations, based on 53 different relation types, out of which the top 5 were: *job_title*, *visited*, *birth_place*, *associate*, *birth_year*.

Despite lacking topics and relevance judgments, this dataset provided sufficient information for initial tests with graph-based retrieval models at a small scale. It only required 980 KiB of storage space, making it easily manageable, and it easily provided an indexable combined data collection that we could issue queries over.

### 4.3.2 BBC Dataset

The BBC dataset[1] [278] contains BBC news from 2004 and 2005, distributed among five categories: *business*, *entertainment*, *politics*, *sport*, and *tech*. There are 2,225 news articles in total, with each category containing between 386 and 511 plain text documents, each ranging between 89 and 4,432 words. This dataset was used to train a query classifier based on these five news article categories.

1 http://mlg.ucd.ie/datasets/bbc.html

## SUMMARY

In this chapter, we presented the datasets that supported or were created during this doctoral work. We used different levels of detail, depending on the importance of the dataset to our work. In particular, we presented test collections used to evaluate information retrieval tasks, focusing on describing the INEX 2009 Wikipedia collection, which is the main dataset that we used for evaluating entity-oriented search tasks, based on a general retrieval model, and TREC Washington Post Corpus, which is a more recent and promising dataset with surrounding work tackling similar tasks. We then described contributed datasets, covering the Simple English Wikipedia Link Graph with Clickstream Transitions 2018-12, which we prepared for the small-scale evaluation of link analysis algorithms. Finally, we described other datasets that we have used during this work, usually with auxiliary or secondary functions, to support the development of our evaluation framework, Army ANT, and our initial implementations of retrieval models.

# 5 SOFTWARE

## Contents



During this doctoral work, we produced two main software artifacts, ANT and Army ANT. ANT is an entity-oriented search engine that we developed as prior work leading to this thesis, where we learned about entity-oriented search and the challenges in the area. In particular, it became evident that there was no clear way to integrate corpora and knowledge bases during indexing, without losing information. On the other hand, integrating signals during ranking, as computed based on each representation model, did not offer a way to cross-reference information from text and entities. This provided focus to our work, on one side supporting the pursuit of a joint representation model for text, entities and their relations, and on the other side presenting an opportunity to explore a more general approach to retrieval, with the creation of a universal ranking function for multiple entity-oriented search tasks. Army ANT was created as an evaluation platform built to support such innovation in entity-oriented search, centered around the indexing of the extended documents introduced in Chapter 4 and illustrated in Figure 4.1. Our aim was to provide an environment where combined data could be loaded into a common data structure, so that joint representation models could be created, and novel ranking functions based on these models could be evaluated and compared to classical baselines.

The structure of this chapter is organized as follows:

- **Section 5.1** describes the ANT search engine, its system architecture [§5.1.1], the event ranking approach used to select the top three events presented in the landing page [§5.1.2], and the query understanding system used to segment and classify queries that were then rewritten for improved performance, based on the identified semantic information [§5.1.3].
- **Section 5.2** describes Army ANT, documenting its system architecture [§5.2.3], command line and web interfaces [§5.2.4], and describing a typical workflow, supported on a simple use case [§5.2.5].

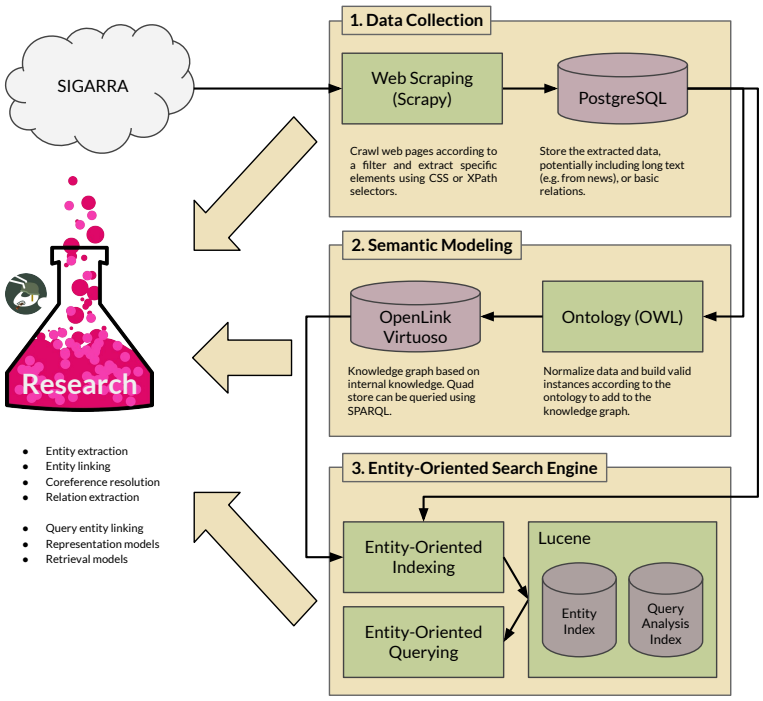



**Figure 5.1:** ANT system architecture: overview.

## 5.1 ANT: ENTITY-ORIENTED SEARCH AT THE UNIVERSITY OF PORTO

ANT[1] is an entity-oriented search engine and research prototype that indexes entities from SIGARRA, the information system at the University of Porto. While the development of the search engine technology, at the backend level, was our full responsibility, ANT also benefitted from the contributions of several postgraduate students[2], who worked on the design and development of the web interface, provided user experience studies, and contributed towards the unofficial mobile application. The search engine indexes seven entity types: *#student*, *#staff*, *#room*, *#department*, *#programs*, *#courses*, and *#news*. It also supports the five query categories proposed by Pound et al. [3]: ENTITY, TYPE, ATTRIBUTE, RELATION, and KEYWORD. While *#news* entities could have been indexed as documents, the search engine purely provides entity search, rewriting the query based on the detected category, in order to improve effectiveness. It also provides an attribute decorator, that directly fetches entity attributes represented by literals in the knowledge base, and a relationship decorator, that fetches entities and attributes common to the entities in a RELATION query.

In the following sections, we describe the system architecture, based on the combination of an inverted index and a triplestore, the ranking approach for the selection of events displayed in the landing page, and the query understanding strategy that we implemented to take advantage of available linked data, originating from information extracted from SIGARRA.

### 5.1.1 System architecture

The ANT search engine follows a simple high-level architecture, as shown in Figure 5.1. It begins by collecting and extracting data from the web, which is initially stored in a relational database. This data is then parsed into triples, following sim-

1 https://ant.fe.up.pt/
2 Visit https://ant.fe.up.pt/about for detailed information on the ANT's contributors.

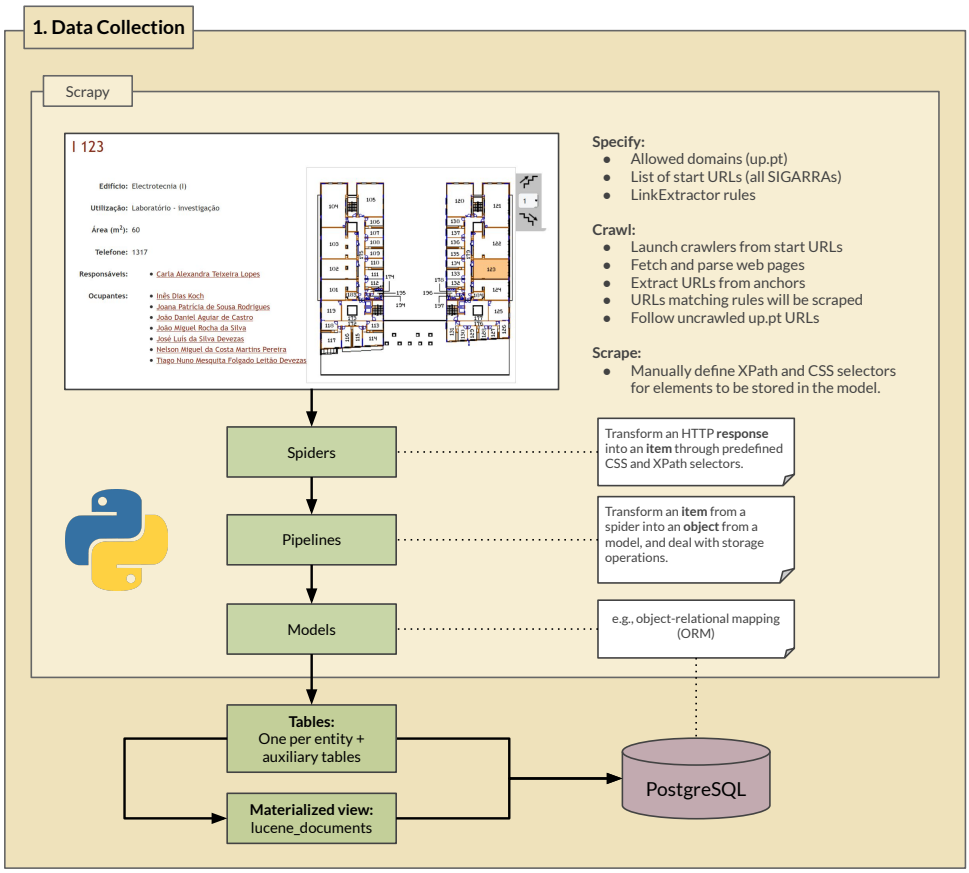



**Figure 5.2:** ANT system architecture: data collection.

ple rules based on a custom ontology that we designed[1]. Albeit internal to the collection (i.e., no external information is added), this source of linked data results in a knowledge base capable of providing supporting information to tasks like event ranking (Section 5.1.2) and query understanding (Section 5.1.3). Finally, we issue an SQL query for each entity type, in order to build a document that can be added to the entity index, and we issue a SPARQL query from the union of basic entity attributes that are common to all types of entities, in order to build the auxiliary query analysis index.

**DATA COLLECTION** Figure 5.2 illustrates the data collection approach based on the *Scrapy* Python library[2]. We start from a SIGARRA web page — in the figure, we use the page for room *I123* as an example — and we define a spider per entity type, where we configure the allowed domain, the list of start URLs, and the link extraction rules based on regular expressions matching relevant links to process or traverse. When a web page is matched for processing, a spider uses CSS and/or XPath selectors to extract specific elements that are then instanced as an object of the model and stored in the relational database through the pipeline. A single table is usually defined for representing one entity type, with a few auxiliary tables used to model secondary elements, like the different professional positions available to staff members. We rely on a common JSON schema to represent the entities that will be displayed in the frontend. In order to improve performance, we pre-cache this data using a materialized view, usually updated during indexing, given that entities only become searchable at that time.

1 https://ant.fe.up.pt/ontology/ant.owl
2 https://scrapy.org/

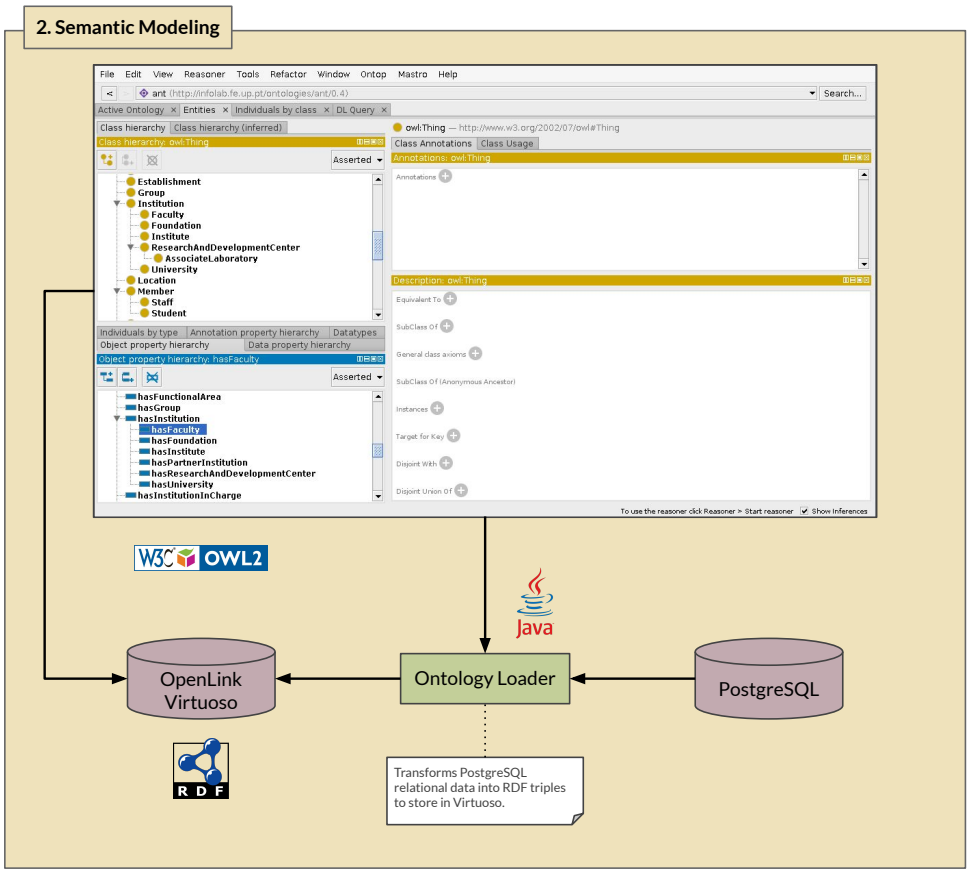



**Figure 5.3:** ANT system architecture: semantic modeling.

**SEMANTIC MODELING** Figure 5.3 illustrates the semantic modeling stage, where we rely on the defined ontology — depicted by the corresponding OWL opened in Protégé[1] — to establish complex relations between the data stored in the relational database. This transformation results in instances of a given class (i.e., entities) linked by object properties among themselves (entity-entity relations), as well as literals (i.e., attributes) linked by data properties to the entities they describe (entity-attribute relations). The process requires issuing SQL queries to fetch the flattened relational data, which is then filtered and selected in order to be converted into semantic triples, that are stored in a quad store.

**ENTITY-ORIENTED SEARCH ENGINE** Figure 5.4 describes the core module of the search engine, where indexing and querying are detailed.

**Entity-oriented indexing** As we can see, we rely on four low-level indexes to query the text, entities and relations that we collected. The entity index contains entity metadata, based on a selection of attributes from the relational database. The query analysis index contains fields for entity label, entity URI, type URI (i.e., the class), and associated semantic category, for supporting query understanding. Both of these indexes are inverted files created using Apache Lucene. This means that they are represented through a finite state transducer that stores inverted document frequencies for terms[2], and a skip list to help navigate and intersect the postings[3], which contain statistics like term frequency, or the positions of the term, for each document containing it. Our semantic index was, in practice, abstracted by our triplestore, OpenLink Virtuoso, which was supported by a SPOC index and a POSC

1 https://protege.stanford.edu/
2 https://issues.apache.org/jira/browse/LUCENE-2792
3 https://issues.apache.org/jira/browse/LUCENE-866

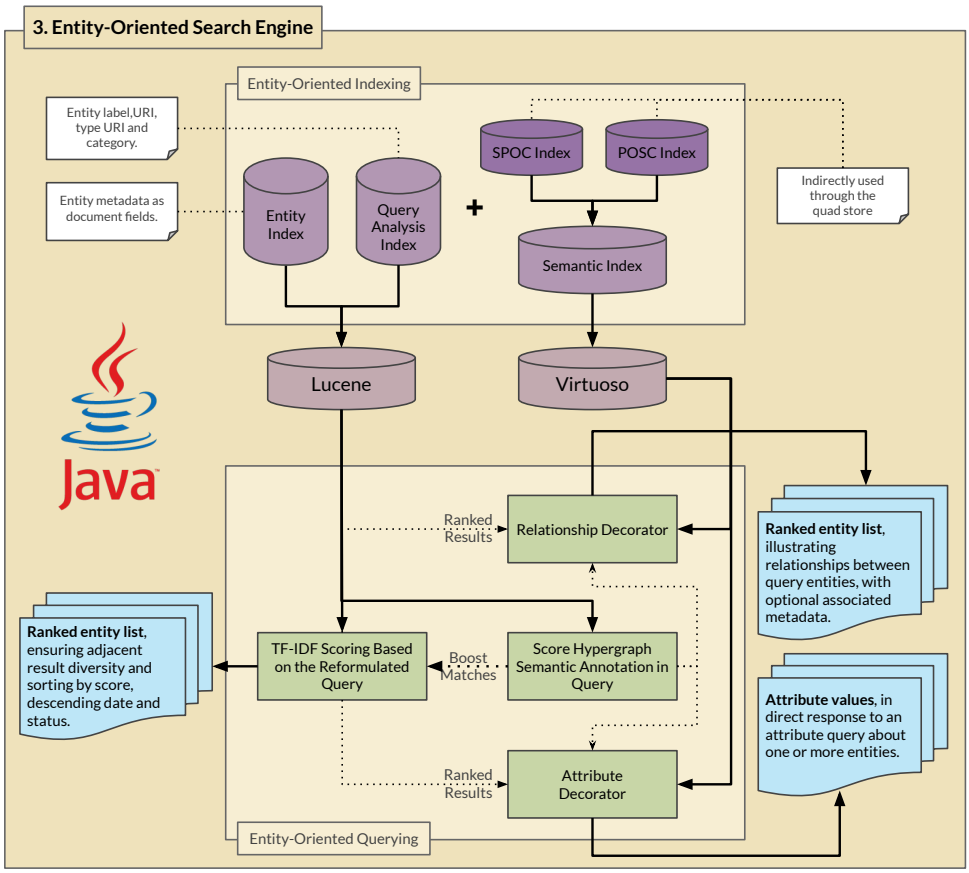



**Figure 5.4:** ANT system architecture: entity-oriented search.

index, where 'S' stands for subject, 'P' stands for predicate, 'O' stands for object and 'C' stands for context (sometimes equivalently replaced by 'G', which stands for graph). A Lucene index can be queried through its own syntax that supports field constraints, wildcards, term boosting, boolean operators, grouping, and field grouping, as well as fuzzy, proximity, and range search[1]. On the other hand, the semantic index, OpenLink Virtuoso, can be queried through SPARQL.

**Entity-oriented querying** While we are required to sequentially query both Lucene indexes for each search, we only query the semantic index to create the query analysis index (offline), or to obtain an attribute or relationship decorator (online). The final ranking is provided by the entity index, based on a query that is formulated according to a set of rules supported on the output of the score hypergraph, as described in Section 5.1.3. Depending on the query category identified during the semantic tagging step, one or both of the decorators might be run. For a relation query, a list of entity identifiers, taken from the top $n$ results, with $n$ being the number of identified entities in the query, is passed to the relationship decorator. Similarly, for an attribute query, a list of entity identifiers, as well as a list of attribute identifiers is passed to the attribute decorator. Each decorator builds a SPARQL query from the provided identifiers, that either returns a list of entities that establish a neighboring connection between the input entities (e.g., a shared department), or the requested attributes for the input entities (e.g., the room and department for two staff members).

A REST API is provided for multiple services, covering: entity search, semantic decorators, search query and result click logging, query autocompletion (experi-

1 https://lucene.apache.org/core/2_9_4/queryparsersyntax.html

mental), and analytics. An API console[1] is also available through the OpenAPI specification from Swagger[2].

### 5.1.2 Information extraction for event ranking

Combining corpora and knowledge bases is frequently used to improve search results, but it can also be used for related tasks, like query-independent event ranking. In order to support this task, we built a complete information extraction pipeline for the Portuguese language, covering all stages from data acquisition to knowledge base population. Accordingly, in this section, we describe a practical application of the automatically extracted information, to support the ranking of upcoming events displayed in the landing page of the institutional search engine, where space is limited to only three relevant events. The approach consists of manually annotating a dataset of news from SIGARRA, covering event announcements from multiple faculties and organic units of the institution. We use this datasets to train and evaluate a named entity recognition module. We then rank events by taking advantage of identified entities, as well as *:partOf* relations, in order to compute an entity popularity score, as well as an entity click score based on implicit feedback from clicks within the institutional search engine. We combine these two scores with the number of days to the event, obtaining a final ranking for the three most relevant upcoming events.

In particular, the goal is to rank news that cover event announcements, in order to display the three most relevant upcoming events of general interest to the local academic community. This is illustrated in Figure 5.5, where we show the landing page for the ANT search engine, highlighting the upcoming events widget. Our approach to event relevance scoring is based on three factors: (i) the overall popularity of the mentioned entities, (ii) the overall popularity of the mentioned entities from clicked event news, and (iii) the number of days left until the event starts. Despite lacking an innovative facet, we present a complete implementation of an information extraction pipeline, for the Portuguese language, describing the whole process from data acquisition to the final application, where we solve a ranking problem from a real-world search engine with the knowledge base we automatically constructed.

#### 5.1.2.1 *An overview on information extraction*

Information extraction consists of uncovering information from unstructured data. While it can be applied to different types of data, we only focus on textual content. Natural language expressed as text is an extremely rich source of information and the most common means of sharing knowledge among humans. Machines, however, rely on the identification of structure within this kind of unstructured data in order to be able to find answers to increasingly complex questions. Therein lies the goal of information extraction. Given a text, a typical pipeline would consist of named entity recognition (e.g., *José Devezas* or *InfoLab*) and classification (e.g., *#person* or *#Organization*), followed by the extraction of relations between identified entities, usually based on patterns centered around verbs (e.g., *:worksAt*). So, given a text with the sentence "José Devezas works at InfoLab.", an information extraction pipeline would be able to build a machine-understandable statement for the ⟨*#Person*, *"works at"*, *#Organization*⟩ relation, identifying *José Devezas* as a *#Person*, *InfoLab* as an *#Organization* and *:worksAt* as the relation between *José Devezas* and *InfoLab*. This type of relations is usually stored in a triplestore. A triplestore contains statements (or triples) in the format ⟨*subject*, *:predicate*, *object*⟩, which, as a whole, form a knowledge base. The knowledge base is a semantic network that

1 https://ant.fe.up.pt/api-console/
2 https://swagger.io/

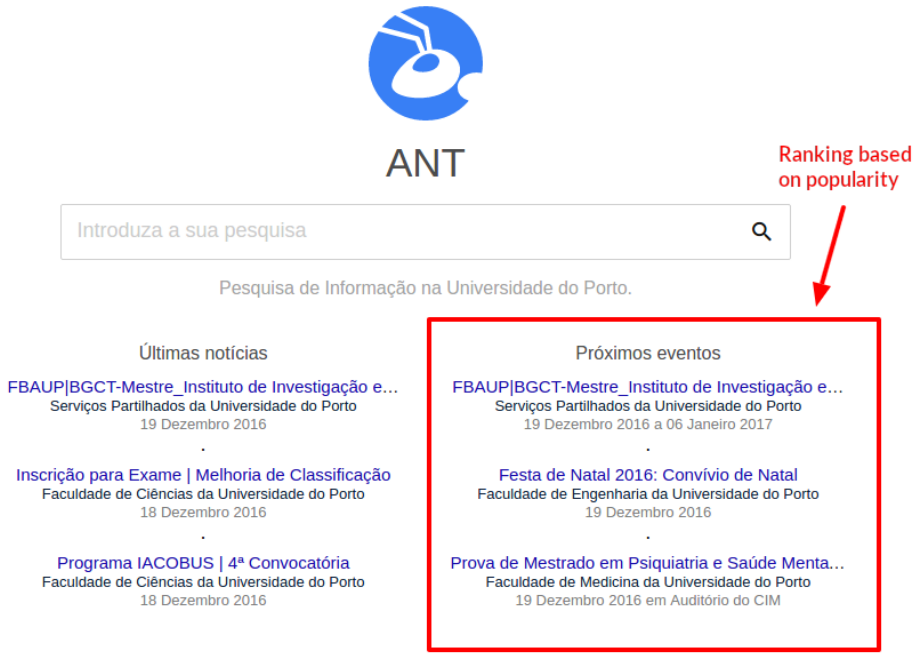



**Figure 5.5:** ANT: entity-oriented search engine for the University of Porto. Upcoming events widget is highlighted, illustrating the target application of the entity-based ranking strategy.

links concepts and provides a structured way of accessing information — individual statements represent information that, when combined, can provide knowledge.

Information extraction can be classified as closed or open, regarding the domain of application. When the domain is closed, we can easily take advantage of hand-crafted rules or gazetteers (dictionaries of entities) in order to link chunks of words to a particular entity, via a regular expression or through string matching, respectively. Open information extraction is used when the domain is unknown or too extensive to cover manually, as is the case of the web. Banko et al. [107] focus on the extraction of relational tuples from the web, without any human input, taking into account issues like automation, corpus heterogeneity and efficiency. They present *TextRunner*, which uses a self-supervised learner to measure the trustworthiness of candidate tuples, along with a single-pass extractor to identify each candidate tuple, using a redundancy-based assessor module to decide on which trustworthy tuples to keep, with a given probability. Their system is also able to be queried, in a distributed manner, supporting complex relational queries that go beyond a traditional search engine.

The typical result of an information extraction pipeline is a knowledge base. Google is currently supported on their own well-curated knowledge base, Knowledge Graph, which evolved from Freebase. However, while perhaps less well-known, they are also working on Knowledge Vault [279], an automated approach to build the automated equivalent of Knowledge Graph. Knowledge Vault does information fusion based on supervised learning and prior knowledge derived from existing knowledge bases, associating a probability with each extracted statement. Their goal is to enable question answering and entity-oriented search systems, with fewer dependence on manual labor.

Having built such a knowledge base, and in order to support entity-oriented tasks, entity linking [227] is usually required. First, there is the need to identify entities in a text, for instance with resource to techniques like Conditional Random Fields [280], which is the approach we use in our pipeline. Then, named entity disambiguation is usually required to be able to identify the entity associated

with a given word or sequence of words. Nebhi [281] proposed a named entity disambiguation strategy based on entity popularity and syntactic features, trained using a Support-Vector Machine (SVM) classifier. Entities extracted from text were matched with entities in a knowledge base (in this case Freebase was used) and the disambiguation module determined which candidate entity was more likely to be associated with that particular textual representation. Entity popularity was used as fallback, while syntactic features were responsible for providing richer indicators than a bag-of-words approach, being able to capture dependencies between words in a sentence and, in a way, providing additional context. For our event ranking approach, we rely on scores computed over a query analysis index, which are similar to entity popularity.

**TOOLS** There are several tools for natural language processing, implementing named entity recognition, as well as relation extraction. The tool we decided to use was the Natural Language ToolKit (NLTK) [282]. NLTK is a Python library that integrates well with other tools such as MaltParser [283], a dependency parser written in Java, or Stanford NER[1], a named entity recognition Java command line tool and library, based on Conditional Random Fields (CRFs). NLTK directly provides access to corpora in multiple languages, to train for instance a POS tagger. POS tagging can be used as a feature for named entity recognition or relation extraction. We use Stanford NER without POS tags to extract entities, but build regular expressions based on words and their POS tags to extract relations.

In order to learn a CRF for the Portuguese language, able to support a custom set of entity types, we first needed to annotate our own dataset. One of the easiest ways to annotate a dataset is to use BRAT (BRAT Rapid Annotation Tool) [284]. BRAT is a web application, built in Python, that requires little configuration and a corpus of plain text documents to annotate. We provide additional detail on the configuration and usage of BRAT in Section 5.1.2.3.

Other relevant tools include LemPORT [285], a lemmatizer for the Portuguese language. Lemmatization can be useful for instance to resolve multiple verbal forms of the same verb into a single word, enabling better relation extraction. This is less efficient but more effective than stemming, where the suffix of the word is trimmed with the goal of obtaining common prefixes for similar words. Also regarding the Portuguese language, a well-known tool is REMBRANDT [286], a framework for semantic information extraction, trained with the Second HAREM Portuguese corpus [287] and achieving first place in the HAREM evaluation forum, with an F-measure of 0.625. Finally, for the English language, there is also GATE [288], a General Architecture for Text Engineering, and, in particular Mímir [289], a multi-paradigm indexing and retrieval framework, which is capable of searching over heterogeneous data sources, like text, annotations, ontologies and knowledge bases. GATE Mímir shares many of the goals of the ANT search engine, combining information extraction and retrieval in a single framework.

**CORPORA** There are at least two Portuguese corpora, available through NLTK, with annotated POS tags: Mac-Morpho [290] and Floresta [291]. These datasets are also called treebanks, which are corpora annotated with syntactic or semantic sentence structure. In this particular work, we only rely on Floresta treebank to train a POS tagger based on its *(word, tag)* tuples. The POS tags can be useful for named entity recognition and for relation extraction, but we only use this feature during relation extraction to build regular expressions.

While there are several tools for the English language, with pre-trained models for multiple tasks, the Portuguese language has been less explored, despite having its own particular challenges. These include dealing with European and Brazilian Portuguese, or even capturing the changes from the latest Portuguese orthography

1 http://nlp.stanford.edu/software/CRF-NER.shtml

reform (1990)[1]. More importantly, the lack of resources for the Portuguese language is of great concern. One of the main, or perhaps the only publicly available dataset with annotated entities and relations for the Portuguese language is HAREM [287]. There are two versions of this dataset, but only the second version includes annotations regarding semantic relations between entities. While there are other datasets, their availability is a challenge. One way to provide a larger amount of annotated data is simply to push for the semantic web and to motivate people to annotate their web sites using RDFa or Microdata. Were this to happen, it would represent a valuable resource for the information extraction community. For now, however, the best approach is to either annotate our own dataset or to use HAREM. In this work, we describe a way to easily annotate our own dataset, with a set of entity types of our choice, based on BRAT.

#### 5.1.2.2 *Approaching event ranking*

We rank academic events based on the news articles that announce them. In particular, we use an information extraction pipeline for named entity recognition, as well as relation extraction, over institutional news. We then build a knowledge base, containing event information, but also *:partOf* relations between pairs of organizations and locations. This enables us to rank events based on historical information on entity popularity and news article clicks, while also propagating popularity through *:partOf* relations. It means that our ranking function captures the following information: if events about *Information Retrieval* are popular and events at *Faculty of Engineering of the University of Porto* are also popular, then an event at *Department of Informatics Engineering* about *Information Retrieval* will have a high probability of being relevant to the community and should have a higher rank. While the link to *Information Retrieval* was directly established, *Department of Informatics Engineering* indirectly received a high popularity weight because it is *:partOf Faculty of Engineering of the University of Porto*.

Since our system does not do entity disambiguation, whenever the same department naming is used across different universities, popularity is be propagated through all *:partOf* links. In order to improve this, entity disambiguation would be a solution, but we could also assign an uncertainty weight proportional to the number of target nodes in the *:partOf* relation, similar to IDF — the more target nodes would be reached, the less reliable the information would be. For our particular domain, however, we predict this to be of low impact — in fact, considering this type of relations without disambiguation might be a way of avoiding popular venues to control the ranks, giving a chance to similar events, happening at different venues, of receiving a higher rank.

#### 5.1.2.3 *Data acquisition, model training and evaluation*

The ANT search engine periodically collects and processes data from SIGARRA, the information system at the University of Porto. As we have seen in Section 5.1.1, data is extracted using a combination of XPath and CSS selectors and stored in a relational database. While most of the data, like student or staff profiles, is structured, there is also some unstructured data from SIGARRA news. We are particularly interested in extracting information about events, locked within SIGARRA news as natural language text. News announcing events can be identified based on whether there is an associated event start or end date in the database. We only retrieve and process news articles with at least one event date associated and skip articles in languages other than Portuguese. The complete name entity recognition approach, from data acquisition to model training and evaluation, is described as follows:

1. Query the relational database for a subset of news articles that announce events, and store them as a CSV file (one article per row).

1 http://www.portaldalinguaportuguesa.org/?action=acordo&version=1990

**Table 5.1:** Entity types used to annotate the SIGARRA News Corpus, along with examples aligned with Figure 5.6. Listed entity types were used to populate the `[entities]` section of the *annotations.conf* file in BRAT.

| Entity Type | Example |
|---|---|
| *#Person* | *"Liliana de Jesus Duarte da Mota"* |
| *#Organization* | *"Faculdade de Direito da Universidade do Porto"* |
| *#Event* | *"Provas de Mestrado em Direito - Licenciada Liliana de Jesus Duarte da Mota"* |
| *#EventType* | *"Provas de Mestrado"* |
| *#Topic* | *"Direito"* |
| *#Location* | *"sala 228"* |
| *#Date* | *"16 de dezembro de 2016"* |
| *#Time* | *"11h00"* |

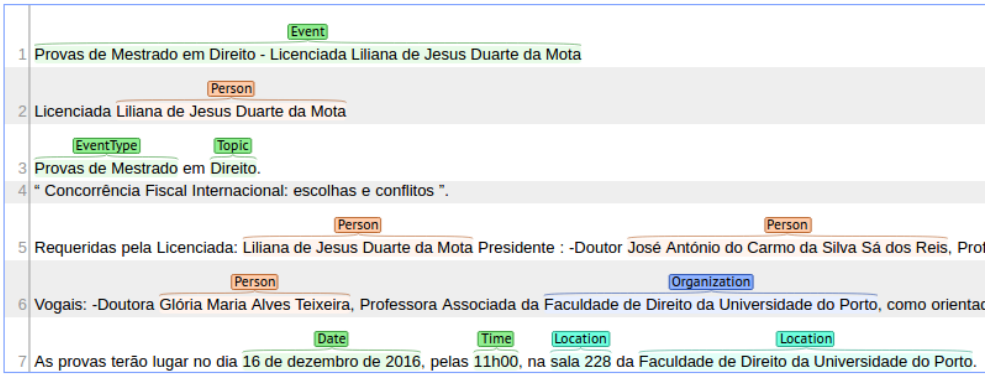



**Figure 5.6:** Example of an annotated fragment of text, for a SIGARRA news, in the BRAT Rapid Annotation Tool.

We retrieved the news articles for the 1,000 closest upcoming events, starting from December 23, 2016, and containing contributions from all faculties and organic units of the University of Porto.

2. For each row in the CSV, prepare a *txt* file containing title, subtitle and content, as well as an empty *ann* file.

   SIGARRA news articles communicate a lot of relevant information in the title and subtitle. In particular, we used the title to extract *#Event* entities and the subtitle to extract information like the location or the date of the event. The content was provided as an HTML fragment that we converted into text using `bs4.BeautifulSoup`, after removing `<script>` and `<style>` tags.

3. Place individual files within a subdirectory of the *data/* directory from the BRAT Rapid Annotation Tool and create an *annotations.conf* file with the list of entity types to annotate (shown in Table 5.1, along with examples).

4. Run `./standalone.py` in BRAT's root directory and manually annotate the corpus, as shown in Figure 5.6 (25 out of the 1,000 retrieved documents were annotated).

5. Split the annotated corpus into two separate directories, one for training (70%; 18 news articles) and another one for testing (30%; 7 news articles).

   While the size of the annotated corpus was not ideal, surprisingly 18 annotated documents were enough to build a working system. Additionally, the annotated corpus could be grown over time, in order to retrain and re-evaluate the system.

6. Convert training documents into a single *col* file (tab-separated format supported by Stanford NER), and testing documents into individual *col* files to enable per-document evaluation.

7. Train a CRF using Stanford NER command line interface and obtain a model for Named Entity Recognition (NER).

**Table 5.2:** Evaluation of the named entity recognition module, based on the test set for the SIGARRA News Corpus.

| Avg. Precision | Avg. Recall | Macro F1 |
|---|---|---|
| 0.633220 | 0.371867 | 0.468563 |

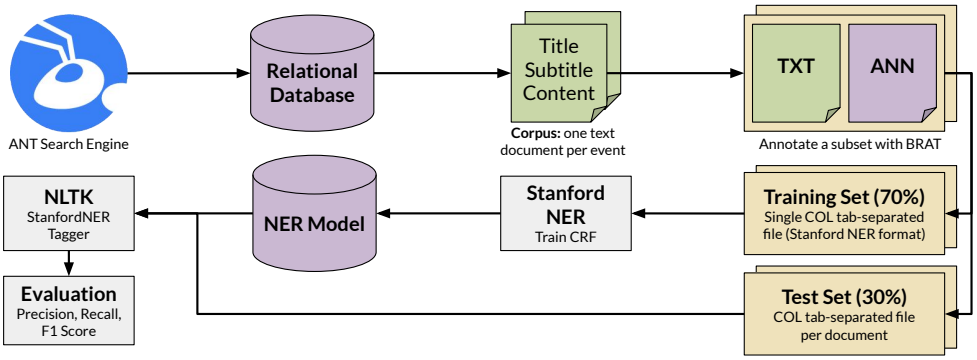



**Figure 5.7:** Data acquisition, named entity recognition and evaluation. Data is obtained from the ANT search engine database and pre-processed to enable CRF training and testing.

8. Evaluate the NER module. Extract entities from the original, non-annotated documents of the test set, using *StanfordNERTagger* from NLTK along with the learned CRF model. Compare extracted entities with annotated entities based on the *col* files, computing precision, recall and F-score.

   There are several evaluation methods for NER [292]. In this particular case, we used the CoNLL evaluation scheme, where only exact matches are considered as true positives. We calculate the precision and recall for each document and then the overall averages for the test set. We also compute the macro-averaged F1-score.

Figure 5.7 shows a complete overview of the named entity recognition module, from data acquisition to evaluation. As we can see, starting from the ANT search engine, we access the relational database, in order to obtain a corpus, initially stored as a CSV file. We then convert each document into a text file, with an associated empty annotations file. The corpus is annotated using BRAT and then split into a training set and a test set. The training set is used to train a CRF[1], obtaining a NER model. The NER model is then used by the *StanfordNERTagger* from NLTK to evaluate the effectiveness of the named entity recognition module based on the test set. Evaluation results are shown in Table 5.2 — we obtained an average precision of 63%, which is acceptable given the reduced size of the annotated corpus, as well an average recall of 37% and a macro-averaged F1-score of 47%.

#### 5.1.2.4 *A pipeline for information extraction*

The complete information extraction pipeline consists of querying the relational database, iteratively processing each individual news article, obtaining a set of triples based on the identified entities and relations from the document, and loading them into the triplestore. The pipeline we built was based on NLTK and it includes the following steps, that are applied to each text sequentially:

**DETECT_LANGUAGE()** Our pipeline was trained using a Portuguese corpus. In order to ensure we only process Portuguese documents, we use the Python wrapper for the well-known `langdetect` Java library[2]. Whenever a text is not

1 We used a default configuration for Stanford NER, based on the example in the FAQ: http://nlp.stanford.edu/software/crf-faq.shtml#a.

2 https://github.com/Mimino666/langdetect

identified as *"pt"*, it is simply skipped. The associated event might still be featured in ANT's event widget, but only when it's extremely close to the date and no other, more relevant events, are held simultaneously.

**SEGMENT_SENTENCES()** The first step in processing a SIGARRA news article is to split the text into individual sentences. We used the pre-trained Portuguese model for the Punkt sentence tokenizer provided by NLTK. Challenges associated with this task are frequently associated with distinguishing actual sentence ends from periods that do not end a sentence (e.g., 'Mr.' or 'Dr.'). While this is a task that is essentially solved and already achieves a high precision in the state of the art, given the quality of our text, many times sentences were incorrectly identified. However, we did not find this to be particularly impactful in the end result, as we did not rely too much on syntactic or dependency parsing or even on POS tags, which we only used for relation extraction.

**TOKENIZE_SENTENCES()** Given a list of strings, obtained from the previous step and corresponding to a sentence each, we split each string into individual words, as well as punctuation. For this, we used `WordPunctTokenizer` from NLTK, instead of the default tokenizer implemented in `nltk.tokenize.word_tokenize`, which was `TreebankWordTokenizer`. While the default tokenizer reliably identified words, without splitting for instance compound words by dashes, it did not implement the `span_tokenize()` method, which was fundamental to simplify the conversion process from the BRAT standoff format (`ann`) into the Stanford NER tab-separated format (`col`). During tokenization, we also replaced any slash characters by a dash, since slashes are removed by *StanfordNERTagger* and, as we explain next, we need to match tagged sentences, word by word, in order to build a tree combining POS tags and named entity tags.

**POS_TAG_SENTENCES()** Given a list of lists of words and punctuation, from the previous step, we assigned a POS tag to each word, obtaining a list of lists of *(word, pos_tag)* tuples. Since there is no pre-trained model for POS tagging Portuguese sentences in NLTK, we used the provided Floresta treebank corpus to learn our own model. For training, we used a `nltk.BigramTagger`, falling back to a `nltk.UnigramTagger` and then to a `nltk.DefaultTagger`, which always identifies a word as a noun (the most common case), when all else fails. An evaluation of a similar POS tagger based on Floresta treebank was already available[1], showing an accuracy of 89%, 87% and 18%, respectively for each tagger.

**NE_TAG_SENTENCES()** Given a list of lists of words and punctuation (without POS tags), we assigned a named entity to each word, obtaining a list of lists of *(word, ne_tag)* tuples. In order to do this, we used the model we had trained for Stanford NER, feeding it to the *StanfordNERTagger* (see step 8 in Section 5.1.2.3 for more details).

**BUILD_SENTENCE_TREES()** Given the lists generated by `pos_tag_sentences()` and `ne_tag_sentences()`, we generated a list of `nltk.tree.Tree` (one per sentence). In order to do this, we used the `nltk.chunk.util.conlltags2tree()` function, which takes a sentence argument as a list of `(word, pos_tag, ne_tag)` tuples. In order to build such tuples, we assumed that an exact equivalence could be established between the two lists, otherwise the system would fail, raising an exception, which skipped the news article. The resulting tree had three levels, a root level (the sentence), a mid-level (the entity types) and a bottom-level (leafs corresponding to chunks of words belonging to a named entity, as aggregated by the mid-level nodes).

1 http://www.nltk.org/howto/portuguese_en.html

**EXTRACT_ENTITIES()** In order to extract the entities, we simply iterated over the trees constructed in the previous step, aggregating tokens per entity type. This is one of the outputs of our system, which is mainly used for evaluation purposes, regarding the effectiveness of the named entity recognition module. Results were saved as a human-readable *ent* text file (one per document), containing lists of entities, grouped by type, with the frequency of the entity in the document. We also saved the same information as a *col* file, in order to compare predicted entities with the ground truth established in the test set. Results have already been shown in Table 5.2 and commented at the end of Section 5.1.2.3.

**EXTRACT_RELATIONS()** Relation extraction consisted of identifying links between pairs of entities. In order to do this, we defined a set of rules, using regular expressions, to identify relations between two types of entities, based on `nltk.sem.extract_rels()`. We then defined an associated list of rules to map the extracted relations into one or more triples in the knowledge base, possibly in reverse order. For example, the rule $\langle$*#Location*, *(da|do)/n*, *#Location*$\rangle$ was used to identify *:partOf* relations between two locations. Each *#Location* entity was then mapped to a *#dul:Place* class, from the DOLCE+DnS Ultralite (DUL) ontology, and the *:partOf* relation was mapped to the *dul:isPartOf* property from the same ontology. Event information was modeled using the Linked Open Descriptions Of Events (LODE) ontology, which focuses on defining a *#lode:Event* class and a set of properties like *lode:atPlace*, that links to *#dul:Place*, or *lode:involvedAgent*, that links to *#dul:Organization* or *#dul:Person*. The Time ontology was also used to describe date and time, using *lode:atTime* to link to a *#time:TemporalEntity* and, in particular, a *#time:Instant* with the property *time:inXSDDateTime* linking to a *#xsd:dateTime* literal.

**BUILD_DEFAULT_RELATIONS()** Since many of the extracted entities were not featured in any relation, and in order to expand our knowledge base to better support the ranking task, we decided to generate some default triples that linked extracted entities to the corresponding *lode:Event*. With this step, we compromised the quality of the information, in the sense that we considered, for instance, all dates as part of the *lode:atTime* relation and we know that some dates were in fact deadlines associated with events. The same is true for locations, as different places were sometimes referenced in news articles, besides the venue of the event. This is something that can be improved over time, either by using different properties like *dul:associatedWith*, or by further developing automatic relation extraction.

**LOAD_RELATIONS_INTO_VIRTUOSO()** Finally, we generate an N-Triples (*nt*) file with the relations identified for each document, including relations of type *:isA*, to annotate each entity with its class. This *nt* file is then loaded into OpenLink Virtuoso, through a POST request to the `/sparql-graph-crud-auth` endpoint[1], storing this information in a separate *ant:EventsKnowledgeBase* graph, using the same naming convention for events and their associated news article, already part of an existing ANT ontology[2].

#### 5.1.2.5 *Entity-oriented event ranking*

By default, ANT simply displayed the three closest upcoming events, without any particular ranking strategy apart from recency. Using a knowledge base, we can go beyond this through entity-oriented approaches. In particular, we propose two scores based on entities associated with events: (i) the entity popularity score,

1 https://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/VirtGraphProtocolCURLExamples
2 Example of URI for *lode:Event* / *ant:NewsArticle*: http://infolab.fe.up.pt/ontologies/ant#Sigarra_News_Article_53369_From_Faculdade_de_Engenharia_da_Universidade_do_Porto.

**Listing 5.1:** SPARQL query to compute $\text{score}_{clk}(e, E_e)$ based on *lode*.

```
1  SELECT ?event ?code ?school (SUM(?count) AS ?score)
2  WHERE {
3    {
4      SELECT ?agent (COUNT(?agent) AS ?count)
5      FROM ant:EventsKnowledgeBase
6      WHERE {
7        ?event a lode:Event .
8        ?event ant:wasClicked "true"^^xsd:boolean .
9        ?event lode:involvedAgent ?agent .
10     }
11     GROUP BY ?agent
12   }
13   ?event a lode:Event .
14   ?event lode:involvedAgent ?involved_agent .
15   ?agent dul:partOf* ?involved_agent .
16   OPTIONAL {
17     ?event ant:hasCode ?code .
18     ?event ant:hasFaculty ?school .
19   }
20 }
21 GROUP BY ?event ?code ?school
```

$\text{score}_{pop}(e, E_e)$, and (ii) the entity click score, $\text{score}_{clk}(e, E_e)$. The entity popularity score is based on the popularity of individual entities, as defined by the frequency an entity appears in distinct news articles over the whole corpus. The entity click score is similar to the popularity score, but is limited to the entities that are linked to clicked event news — this click status is transferred to the knowledge base daily, when the IE pipeline is run, and is stored using the *ant:wasClicked* property, linking to a *#xsd:boolean* literal.

The two scores are computed using a SPARQL query and then stored in the relational database, each in their own column of the news articles table. Events are retrieved by combining the number of days remaining to the event start date, and the entity popularity and click scores. Listing 5.1 shows the SPARQL query used to calculate $\text{score}_{clk}(e, E_e)$ for all *#lode:Event* instances in the system. The $\text{score}_{pop}(e, E_e)$ is calculated using a similar query, where we remove the statement in line 8, discarding the constraint for clicked events. As we can see, each entity score is calculated, per event, based on the total number of links to the entities that are associated with the event. We illustrate this by using *lode:involvedAgent* property for classes *#dul:Person* and *#dul:Organization*.

After computing and storing the two entity scores for each event, we calculate the final score, for event ranking, as shown in Equation 5.1. Given event $e$, and entities $E_e$ associated with event $e$, the final score is calculated based on a weighted average of three factors: days to event $\Delta T_e$, entity popularity score $\text{score}_{pop}(e, E_e)$, and entity click score $\text{score}_{clk}(e, E_e)$.

$$\text{score}(e, E_e) = w_1 \frac{1}{\Delta T_e + 1} + w_2 \frac{\text{score}_{pop}(e, E_e)}{\max_e \text{score}_{pop}(e, E_e)} + w_3 \frac{\text{score}_{clk}(e, E_e)}{\max_e \text{score}_{clk}(e, E_e)} \quad (5.1)$$

We have deployed this ranking strategy in the ANT search engine, manually tuning the weights in order to prioritize the number of days to the event, since close events are more relevant, and then considering the implicit feedback based on the entity click score and finally the entity popularity score. The version we deployed

only relies on entities of type *#Person* and *#Organization* and uses $w_1 = 0.5$, $w_2 = 0.2$ and $w_3 = 0.3$.

### 5.1.3 Index-based semantic tagging for efficient query understanding

The search process begins with the query, making query analysis essential to extract additional information, such as the parts of the query that represent entities, as well as their types or attributes, and the parts of the query that represent traditional keywords. Identifying entities in a query through segmentation, as well as matching them to a particular category is frequently called semantic tagging [293]. In our system, query understanding is fully supported by the information obtained from semantic tagging. This enables the subsequent construction of knowledge base queries to retrieve entities, types or attributes matching the text and identified category of each query part. The resulting ranked set of candidates can then be used to support the understanding of the query, helping in the final query answering process. We evaluate the efficiency of the candidate retrieval subtask, based on a Sesame triplestore, using SPARQL queries, as well as on a Lucene index, prepared for this task, using keyword queries. We also propose the score hypergraph, as an extension and improvement of the probabilistic approach we first consider.

#### 5.1.3.1 *An overview on query understanding*

Pound et al. [3] have provided a relevant contribution to entity-oriented search by structuring the queries for ad hoc object retrieval into five categories: ENTITY query (directly find a specific entity), TYPE query (find entities of a given type), ATTRIBUTE query (find values of an attribute of an entity or type), RELATION query (discover how two or more entities or types are connected) and KEYWORD query (for any traditional full-text query that doesn't fit the other categories).

Guo et al. [294] proposed a new application of named entity recognition in the context of search queries, based on a Weakly Supervised Latent Dirichlet Allocation (WS-LDA) algorithm that used partially labeled entities as seeds. The idea was to use a query log, discovering queries that contained a given entity and class, to obtain an associated context (remaining terms). Based on a context "document" and a class "topic", they generated training data that could be used to learn a topic model and reiterate with new seeds to improve the overall model.

Blanco et al. [295] presented an extremely effective and efficient algorithm for entity linking in queries (Fast Entity Linking, or FEL) that took advantage of context (using word2vec), based on query logs and Wikipedia articles on the entity (as determined by the anchor text linking to the Wikipedia article). While the methodology we present here does not seem to outperform FEL (the mean run time for our whole search process is 49ms for a different dataset), our technique might have a lower implementation cost, as it easily builds on top of existing information retrieval frameworks like Lucene.

Aggarwal and Buitelaar [296] focused on the understanding of natural language queries to facilitate querying over linked data, with languages like SPARQL. Their pipeline included: entity annotation (supported on two indexes, one for labels and URIs of all DBpedia instances and another one for all DBpedia classes), deep linguistic analysis (at this stage, a central entity, as well as the dependencies between all entities, were identified), and semantic similarity/relatedness (similarity was defined on the basis of *:isA* relations of concepts, while relatedness covered other types of relations).

#### 5.1.3.2 *Scaling issues for growing entity data*

ANT is an entity-oriented search system capable of answering queries by taking advantage of the previously untapped underlying structured data available in the current information system. We considered information needs like the discovery of

the department for a given staff member, or finding students enrolled in two given courses. We first tackled this problem at a faculty level and then extended our support to the fourteen schools of the University of Porto. When we scaled from faculty-centric entities to university-centric entities, we identified a performance issue that led us to explore alternatives to directly using the triplestore for query analysis. Growing from a dataset restricted to the students at the Faculty of Engineering to a dataset including the students for all the schools at the University of Porto meant growing our triplestore from 546,760 to 2,594,511 statements. Including the students for the whole university had a tremendous impact in the growth of our dataset, translating into 139,640 additional students, associated with 193,650 extra enrollments, 1,166 additional courses, 14 more academic years and 10 more faculties.

#### 5.1.3.3 *Approaching semantic tagging in queries*

Semantic tagging in queries is the act of annotating queries with entity types, for query understanding. We followed this approach by segmenting the query and annotating groups of sequential terms (n-grams) with the most probable category (ENTITY, ATTRIBUTE, TYPE or KEYWORD), based on a set of matching candidate labels from the knowledge base. In this work, we focused on the efficiency of two alternative methodologies for candidate retrieval, one based on a Sesame triplestore and SPARQL querying, and another one based on a Lucene index and keyword querying. The techniques we describe here can easily be used to also identify entity types or to establish entity links.

The first step for query analysis was to build a collection of all n-grams for $n \in [1, n]$. We used $n = 6$ as the maximum n-gram size, given it provided a coverage of 94.28% for the labels of our entities, resulting in a good compromise between performance and accuracy (a higher number of n-grams would result in additional candidate retrieval queries). The second step was to retrieve matching candidates for each n-gram. We did this either by using the Sesame triplestore or the specialized Lucene index. We also computed the number of candidates per class using either technology. This enabled us to calculate the probability of associating a given candidate to an n-gram: $1 - |C_t^x| \ / \ |C_t|$, where $C_t^x$ is the set of candidates for n-gram $x$ and type $t$, and $C_t$ is the set of candidates for type $t$. The probability is higher when the fraction of candidates is smaller, which means that rarer labels have priority over common labels, resulting in better precision. Finally, in the last step, we selected the n-gram with the highest probability, keeping only the longest n-gram in case of term overlap between selected n-grams. Each candidate could be directly categorized into *entity*, *attribute* or *type*. This information was used to classify the query based on templates for these three categories.

Our first attempt at retrieving matching candidates was directly based on the Sesame triplestore. This contained our knowledge graph and was the obvious choice for an initial approach. As described in Section 5.1.3.2, we first experimented with a knowledge base containing 546,760 statements or facts. While this approach did not allow for sub-second query times, it resulted in a reasonable query time of under 5 seconds. The SPARQL query we built returned four columns associated with candidate entities: *Label*, *URI*, *Class* and *Category*. This was obtained from the union of three sub-queries for *entity*, *attribute* and *type* individuals, associating the value of the property *rdfs:label*, or equivalent, to the *Label* column. These results were filtered using a case insensitive regular expression that matched the n-grams generated from the search query.

As an alternative for better performance, we built an Apache Lucene query analysis index based on the triplestore data, combining documents for *entities*, *attributes* and *types*. Each document contained four fields: *Label*, *URI*, *Class* and *Category*. We iterated through the same items returned by the SPARQL query described above, dropping, however, the regular expression filter. This enabled us to create an index

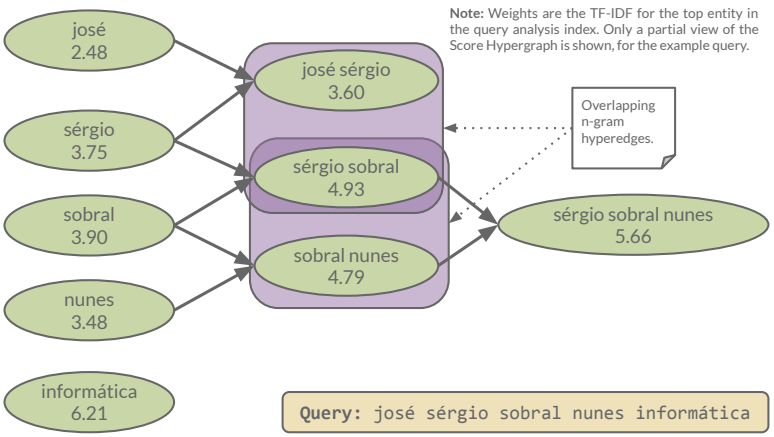



**Figure 5.8:** Score hypergraph: TF-IDF scores for query n-grams based on the query index.

of query parts, as supported by our knowledge base. We then queried this index in order to return the results for each n-gram generated from the search query. We used proximity search within $n = 6$ terms of distance (the same as the n-gram size) and ensured that the query was parsed in order. For each query to the index, we only returned the top-$\mathcal{N}$ results. Specifically we used $\mathcal{N} = 10$, which is a low value that results in high performance, as we show in Section 5.1.3.5.

#### 5.1.3.4 *Solving overlap with the score hypergraph*

The most probable candidate was selected for each query part, however we had to deal with overlap. For instance, it could happen that for query `[ josé sérgio sobral nunes informática ]`, both *"josé sérgio"* and *"sérgio sobral nunes"* would be identified as two of the most probable query parts. This would result in the following query segmentation: `[ josé sérgio | sérgio sobral nunes | informática ]`, which is not a correct result for this query, even if *"josé"* links to an entity containing *"josé sérgio"* with the highest probability.

We propose the score hypergraph, which we developed while thinking about this problem. With the score hypergraph, segmentation is done iteratively, discarding overlapping query parts at each step. While the score hypergraph can be applied to probabilistic scores, it can also be generalized to different weighting schemes. In particular, we applied the score hypergraph based on TF-IDF scores computed from the Lucene query analysis index.

Figure 5.8 illustrates a possible hypergraph generated for the problematic query. Displayed weights were obtained from a production version of the query analysis index, based on TF-IDF. We only show n-grams for $n \in \{1,2,3\}$. We create an edge from each n-gram to the $(n+1)$-grams that contain its label. We also create a hyperedge containing all overlapping n-grams for the same value of n. Each node is weighted according to the TF-IDF of the top ranking result, retrieved when using the n-gram as a keyword query over the query analysis index. The query segmentation process then consists of the following steps:

1. Select the source node (i.e., with lowest n) with the highest weight.
2. Follow edges leading to nodes with a higher weight, until no more are available.
3. Select the end node as a query part.
4. Backtrack through incoming edges, follow hyperedges (without backtracking through their edges), and delete all visited nodes.
5. Repeat from step 1 until the hypergraph is empty.

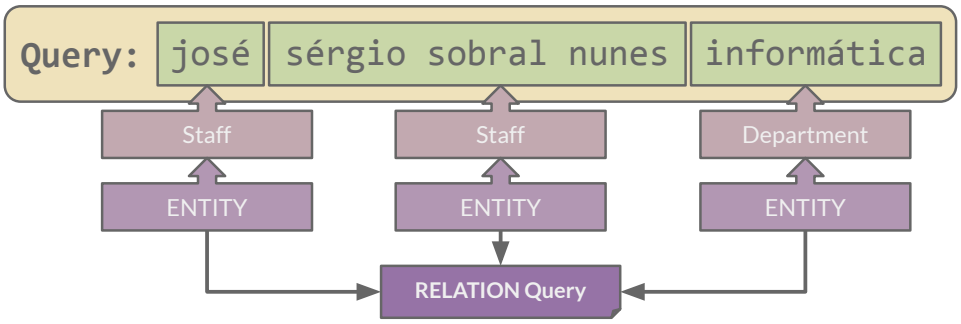



**Figure 5.9:** Score hypergraph: segmented and semantically tagged query.

The top entity associated with the selected query parts, as retrieved from the query analysis index, can also provide a *Class* and a *Category*, which support the semantic tagging, as well as the query classification tasks. This is illustrated in Figure 5.9, where the resulting query segmentation was `[ josé | sérgio sobral nunes | informática ]`. As we can see from the figure, when searching for `[ josé ]` and for `[ sérgio sobral nunes ]`, during candidate retrieval, the top retrieved entities were of class *#ant:Staff* in both cases. When searching for `[ informática ]`, the top retrieved entity was of class *#ant:Department*. While the user's intent when mentioning *"informática"* could have been to search for two staff members from an informatics program, teaching an informatics course, or from the informatics department, the most probable option according to our algorithm would be the latter. Accordingly, the three query parts are of category ENTITY, which ultimately leads to a classification of RELATION query.

Let us consider the following four sets for each category of identified query part: $\mathcal{A}$ for attributes, $\mathcal{T}$ for types, $\mathcal{E}$ for entities, and $\mathcal{R}$ for relations. We implemented the following rule set, applied in order until a match is found, to classify the query according to the five query categories from Pound et al. [3]:

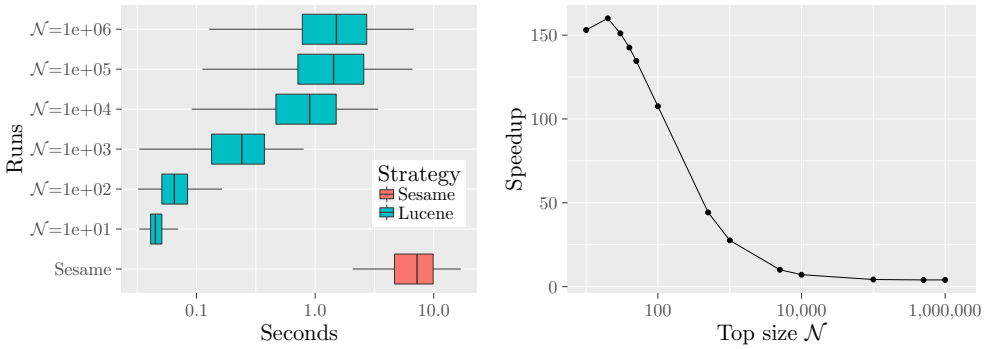
1. $|\mathcal{A}| > 0 \Rightarrow$ ATTRIBUTE
2. $|\mathcal{T}| > 0 \ \wedge |\mathcal{E}| > 0 \Rightarrow$ ATTRIBUTE
3. $|\mathcal{T}| > 0 \ \wedge \mathcal{E} = \emptyset \Rightarrow$ TYPE
4. $|\mathcal{E}| > 1 \Rightarrow$ RELATION
5. $|\mathcal{E}| > 0 \Rightarrow$ ENTITY
6. Otherwise $\Rightarrow$ KEYWORD

#### 5.1.3.5 *Evaluating candidate retrieval efficiency*

We compared the performance of both candidate retrieval strategies by measuring overall search time over a set of synthetic test queries. We synthetically built a query test set by combining terms from randomly selected individuals of the ontology with terms from a Portuguese dictionary with over 400,000 words. Our generation method required five parameters: the number of queries to generate, the minimum and maximum number of terms associated with ontology individuals, and the minimum and maximum number of keyword terms. For this evaluation process, we generated 1,000 queries with the number of terms associated with ontology individuals ranging from 3 to 8, and with a number of keyword terms ranging from 0 to 2, resulting in queries with a minimum of 3 terms and a maximum of 10 terms overall.

**Table 5.3:** Statistics for the query analysis time of the Sesame triplestore and the Lucene index strategies, using $\mathcal{N} = 10$ for the Lucene index.

| | **Sesame triplestore** | **Lucene index** |
|---|---|---|
| **Avg.** | 7.435765s | 0.048580s |
| **Std.** | ±3.206806s | ±0.019115s |
| **Speedup** | 153.062268 (~153 × faster) | |
| **Mann-Whitney U Test** | p-value ≈ 0 ≪ 0.01 | |




**(a)** Run times for the Sesame and Lucene strategies (log scale for the x-axis).

**(b)** Speedup for different values of $\mathcal{N}$ (log scale for the x-axis).

**Figure 5.10:** Efficiency evaluation of the overall search process, based on 1,000 synthetic queries.

**COMPARING QUERY ANALYSIS TIME FOR THE RETRIEVAL STRATEGIES** We did several runs based on the same set of synthetic queries. In particular, we did one run based on the Sesame triplestore strategy, that we directly compared with a run based the Lucene index strategy for the top-$\mathcal{N}$ results. We picked $\mathcal{N} = 10$ since it provided a near-optimal speedup, also having a positive impact on the quality of the results for a small set of manually tested queries.

In Table 5.3, we show the mean query analysis time (Avg.) along with the standard deviation (Std.), in seconds, for the 1,000 synthetically generated test queries. These tests were ran on a laptop with a dual core Intel® Core™ i7-5600U, 16 GiB of RAM and a 256 GiB solid-state drive. We calculated the speedup of the Lucene index strategy over the Sesame triplestore strategy, concluding that it was about 153 times faster, for $\mathcal{N} = 10$. Increasing the parameter $\mathcal{N}$ resulted in lower, but still positive, speedup values, as shown in the next paragraph.

**INFLUENCE OF $\mathcal{N}$ OVER THE SPEEDUP** Figure 5.10a shows a run time comparison between the Sesame strategy (all matching results) and various $\mathcal{N}$ values of the Lucene strategy (top-$\mathcal{N}$ results). As we can see, the index-based strategy outperforms the triplestore strategy even when retrieving the top $\mathcal{N} = 1$ million matching candidates. Figure 5.10b illustrates the evolution of the speedup for growing values of $\mathcal{N}$. Higher values for the parameter $\mathcal{N}$ were expected to result in a lower speedup. However, by analyzing the progression of $\mathcal{N}$, from 10 to 1 million, we found that the speedup actually increased, from $\mathcal{N} = 10$ to $\mathcal{N} = 20$. This can be explained by the fact that our testing routine continuously read from the same location in disk, to load the index before running each set of queries, which resulted in better read performance through system caching. However, as expected, for $\mathcal{N} > 20$, the speedup consistently decreased, nearly stabilizing at 4× faster.

## 5.2 ARMY ANT: A WORKBENCH FOR INNOVATION IN ENTITY-ORIENTED SEARCH

Army ANT is an information retrieval research framework that supports experimentation with classical approaches, while also providing a high-level abstraction that motivates the implementation of innovative approaches in a shared evaluation environment. It facilitates the use of combined data, and it provides an environment for testing and evaluating multiple retrieval tasks, supporting keyword or entity queries, as well as documents, entities, or even terms, as the rankable results.

In this section, we present an overview on Army ANT [§5.2.1] and position it in regard to other experimental frameworks, such as Terrier, Lemur, and Nordlys [§5.2.2]. We then describe its system architecture [§5.2.3], covering four main abstractions: (i) readers, to iterate over text collections, potentially containing associated entities and triples; (ii) engines, that implement indexing and searching approaches, supporting different retrieval tasks and ranking functions; (iii) databases, to store additional document metadata; and (iv) evaluators, to assess retrieval performance for specific tasks and test collections. We also describe the command line interface and the web interface [§5.2.4], presenting learn mode as a way to explore, analyze and understand representation and retrieval models, through tracing, score component visualization, and documentation. Finally, we present a typical workflow of our platform [§5.2.5], describing configuration, server deployment, and the required implementation details for conveniently exploiting learn mode.

### 5.2.1 What is Army ANT?

Army ANT is an experimental workbench, built as a centralized codebase for research work in entity-oriented search. It was created as a structured framework for testing novel retrieval approaches in a comprehensive manner, even when potentially deviating from traditional paradigms. This required a flexible structure, that we developed by iteratively satisfying the requirements of multiple engine implementations for representing and retrieving combined data [92, Definition 2.3]. An important step in research, that we also motivate and support, is the continuous documentation of models and collections, which is fundamental for reproducibility, but also useful to advance research, by exploring, learning and building on previous approaches. In this sense, Army ANT is particularly well suited for individuals or small research teams, as it provides a structural framework with the basic tools to integrate the typical components of a modern search engine. It acts as a skeleton to support the research process, while introducing only minimal limitations, by leaving the storage and retrieval implementations to the researcher.

The basic unit of Army ANT is the engine, which must implement the representation model for indexing and the retrieval model for searching. The indexing method has access to one of multiple collection readers and can optionally consider external features. The search method is based on a keyword query, pagination parameters and, optionally, a task identifier, a ranking function and its parameters, and a debug flag. For searching and evaluating over the web interface, each engine is required to have a unique identifier, which frequently describes the representation model and indexed collection (e.g., *lucene-wapo* for a Lucene index over the TREC Washington Post Corpus). Each engine has an entry in the YAML configuration file (*config.yaml*), so that it is visible to the web interface. Supported ranking functions, their parameter names and specific values can also be defined in the configuration file. Combinations of selected parameter values can then be used by the evaluation module to launch individual runs, known as evaluation tasks. When completed, each task will provide a performance overview, based on efficiency and effectiveness metrics for each parameter configuration, as well as complementary visualizations and a zip archive with intermediate results. Intermediate results in-

clude elements like the average precisions for each topic, used in the calculation of the mean average precision, or the results for each individual topic, along with the relevance per retrieved item, according to a ground truth (e.g., qrels from TREC or INEX). This means that, even if Army ANT evolves and no backward compatibility is maintained, the archive can still be downloaded and independently used to compute other metrics, such as statistical tests, or to correct any wrong calculations. Additionally, an overall table, comparing the performance among different runs, is also available for download as a CSV or LaTeX file.

Out-of-the-box, Army ANT provides reader implementations for INEX 2009 Wikipedia collection [112], TREC Washington Post Corpus[1], and Living Labs API [246]. It also provides a Lucene baseline engine, supporting TF-IDF, BM25 and divergence from randomness, as well as several experimental engines, such as hypergraph-of-entity. Finally, evaluators are available for the INEX Ad Hoc track and the INEX XER track, as well as for the TREC Common Core track and for the Living Labs API's team-draft interleaving online evaluation. On a smaller scale, Army ANT also provides several utility functions, covering DBpedia and Wikidata access, as well as statistics for the measurement of rank concordance and correlation. Several index inspection and debugging tools, as well as documentation strategies, are also integrated into Army ANT's workflow. The workbench is written in Python, providing integrated implementations for engines written in Java and C++, which we use as examples of cross-language interoperability.

### 5.2.2 Frameworks for experimental information retrieval

Over the years, there have been several actively developed frameworks for experimental information retrieval research. In this section, we cover three of them: The Lemur Project, Terrier and Nordlys, the latter being a fairly recent framework in the area of entity-oriented search. Our focus was on comparing the frameworks with Army ANT, however for a more in-depth comparison of open source search engines please refer to Middleton and Baeza-Yates [297].

The Lemur Project [255] is particularly focused on language models. There are multiple components included in Lemur, besides the Lemur toolkit, which has been deprecated in favor of Indri, since version 4.12, released in June 21, 2010. Components include the RankLib, a library for learning-to-rank algorithms, Sifaka, which supports named entity recognition, as well as frequency and co-occurrence analysis, and Galago, which provides a command line and web interface and also supports the computation of PageRank. Indri is the main component of the Lemur Project. It supports structured querying using INQUERY [256] operators and suffix-based wildcards. It also supports field and passage retrieval and the indexing of text annotations and document metadata. Indri can be distributed, both for indexing and retrieval tasks, scaling up to terabyte collections.

Terrier [36] provides easy-to-use out-of-the-box implementations of multiple well-known and state-of-the-art indexing and ranking approaches. It is particularly good for batch evaluation over standard TREC and CLEF collections, providing a command line tool for directly indexing TREC corpora. Terrier is also a good example of a middle-ground framework between research and production, since it not only supports experimental IR, but it is also prepared to be deployed as a product, if used as a Java library.

Both the strength and the downside of the previous two frameworks is that they provide their own implementations within their own ecosystem. This means that, if a certain ranking function is not supported, it must be implemented following the toolkit's interface rules, which frequently aren't clearly documented. This makes novel approaches harder to explore, since the frameworks constrict innovation instead of promoting it. Although, arguably, this is not their main focus.

1 https://trec.nist.gov/data/wapost/

Army ANT attempts to fill this niche and, in order to do so, it is built with the integration of external libraries, such as Apache Lucene, in mind. It can even act as a layer over Lemur and Terrier, since it supports both C++ and Java engines. The challenge in providing a workbench for innovation is that it requires both freedom and structure. For this reason, implementing a new engine in Army ANT is frequently perceived as the equivalent to starting from scratch. However, this is not the case, since Army ANT already provides collection readers and evaluators that can be used directly after implementing a new engine. Also, at the very least, there is already a predefined command line interface and web interface to access the engine, both for indexing the collection and searching over the index. Through the implementation of a class that inherits `Index`, along with its `index()` and `search()` methods, Army ANT provides a skeleton to support research ideas, without being opinionated, but supportive through optional reusability.

Nordlys [298] is a bit different from Lemur and Terrier, as it focuses on three entity-oriented search tasks: entity retrieval, entity linking in queries, and target type identification. Like Army ANT, Nordlys acts as a layer over available information retrieval tools (e.g., Elasticsearch). Unlike Army ANT, Nordlys relies on external evaluation tools, instead of providing a generalized evaluation framework, where different topic or relevance judgment formats are translated to a common representation and assessed through a common module. Army ANT also integrates with novel evaluation frameworks, such as the Living Labs API. While Nordlys is designed for a fixed set of entity-oriented search tasks, Army ANT is able to generalize through a task selection parameter in the `search()` function, which can be freely implemented, or even ignored, by the developer or researcher. Our workbench also supports multiple engines for a single web server instance, enabling the joint evaluation and comparison of potentially different representation and retrieval models. Finally, Army ANT provides a learn mode, that is focused on exploring, understanding and analyzing each engine and, in particular, the ranking functions. In order to achieve this, it relies on the visualization of individual score components, for a given query, per document, and also on the implementation of tracers (Java-only for now), which are essentially explainers of ranking functions that describe the ranking function through a tree of togglable nodes.

#### 5.2.2.1 *Comparing existing frameworks and Army ANT*

Table 5.4 provides a structured comparison of the four frameworks, based on the following features: (1) are they opinionated (i.e., biased towards a particular approach or technology); (2) have they been actively maintained (i.e., relevant commits in the last few months); (3) which test collections are supported; (4) which programming languages and interfaces are supported.

As we can see from Feature 1 ('Opinionated?'), we classified both Lemur and Nordlys as opinionated, since they have a clear focus on language models and entity-oriented search, respectively. We then classified Terrier as non-opinionated, since it focuses on integrating state-of-the-art approaches in a single package. And we classified Army ANT as partially opinionated, since provided implementations are geared towards entity-oriented search, but the framework supports general search tasks based on keyword queries.

Regarding Feature 2 ('Maintained / active?'), we found that frameworks are still actively maintained, although, when Lemur transitioned to Indri, some of Lemur's features, like `ireval` were never integrated with Indri. Both Terrier and Army ANT have shown relevant activity in the last few months, but Nordlys has only received maintenance fixes for its most recent commits.

For Feature 3 ('Suggested test collections.'), we found that all frameworks support TREC test collections, with Terrier also covering CLEF and Army ANT supporting INEX test collections, both for ad hoc document retrieval and ad hoc entity retrieval. Finally, regarding Feature 4 ('Programming languages and interfaces.'), we found

**Table 5.4:** Comparing frameworks for experimental information retrieval.

| Framework | Comment |
|---|---|
| 1. Opinionated? | |
| Lemur | Yes. It has its own query language and, while it supports other weighting models (seen as baselines), it is focused on language models. |
| Terrier | No. Apart from being purely Java-based, it does a fairly good job at keeping its architecture open through plugins and even provides some of the Indri query language operators from The Lemur Project. |
| Nordlys | Yes. It is focused on very specific entity-oriented search tasks. |
| Army ANT | Partially. It supports any retrieval task that uses keyword queries as input, but provided implementations are focused on entity-oriented search tasks. |
| 2. Maintained / active? | |
| Lemur | Partially. The Lemur toolkit has been deprecated, but some of its features (e.g., *ireval*) have not transitioned to Indri. Lemur still runs in Windows, but it won't compile in Linux with recent versions of gcc (6+). The last commit to SourceForge, at the time of this writing, was done in June 2018. |
| Terrier | Yes. New weighting models and several improvements have been added over the years. The last commit to GitHub, at the time of this writing, was done in September 2018. |
| Nordlys | Partially. The last commit to GitHub, at the time of this writing, was done in May 2018, for a bug correction, but the last relevant change occurred almost a year ago. |
| Army ANT | Yes. The last commit to GitHub, at the time of this writing, was done in October 2018, with relevant additions committed in the last month for the rank correlation analysis module. |
| 3. Supported test collections. | |
| Lemur | TREC (*trecweb* for the web format and *trectext*/*trecalt* for the regular format). |
| Terrier | TREC / CLEF (no specific CLEF implementation found). |
| Nordlys | TREC (via `trec_eval` wrapper). |
| Army ANT | TREC (Common Core track and Living Labs API for OpenSearch track) / INEX (Ad Hoc track and XML Entity Ranking track). |
| 4. Programming language and interfaces. | |
| Lemur | C++. Has bindings for Java. Provides a web interface (Galago). |
| Terrier | Java. Provides a web interface (`bin/terrier http`). |
| Nordlys | Python. Provides a RESTful API and a web interface. |
| Army ANT | Python. Provides Java and C++ implementations, integrated into Python through JPype and Boost Python, respectively. Provides a RESTful API and a web interface. |

that every framework provides a web interface, but they are written in different programming languages: Lemur in C++, Terrier in Java, and Nordlys and Army ANT in Python. Lemur also provides Java bindings and Army ANT provides direct integration with Java and C++ engine implementations.

While the frameworks all fit into a similar use case of experimentation and evaluation, some are more adequate for particular tasks or settings. Lemur is interesting to explore language models, through the flexible Indri query language [299], based on INQUERY [256]. Terrier provides a framework for easy TREC-based experiments, where Java classes can be created to act as plugins for specific features, such as a custom weighting model. Multiple state-of-the-art approaches are also provided, along with the flexibility to deploy for production if needed. Nordlys, while still a research prototype, does an excellent job at establishing and providing base implementations for relevant tasks in entity-oriented search. It is currently relies on data from DBpedia [94], FACC [300] and a pre-trained word2vec model based on Google News[1]. These are fundamental features for any entity-oriented modern search engine, that weren't easily available before Nordlys. Finally, Army ANT can be used as a codebase to support novel ideas on information retrieval, without the need to start from scratch in respect to any feature that is secondary to the main idea. For instance, if the focus is on query expansion, then you can simply extend an existing engine, add a query expansion function and feed the expanded query back to the original search method. This is, in a way, similar to what Terrier provides. The difference is that Army ANT's structure is a lot more relaxed and close to the user interaction layer. Nevertheless, even if the main user action is to search, we still need to ensure pagination and a common format for results to be displayed correctly. Once an engine is implemented, it can be immediately used to index any of the supported collections, available through one of the `Reader` subclass, and even to evaluate the retrieval model through standard test collections, as implemented

1 https://github.com/mmihaltz/word2vec-GoogleNews-vectors

through an `Evaluator` subclass. In any case, the developer can implement new readers or evaluators, as required, but the abstractions in Army ANT always remain close to the user requirements.

### 5.2.3 System architecture

Army ants are larger, altruistic ants, that frequently sacrifice themselves for the well-being of the colony, for instance by building a bridge with their bodies to ensure food transportation [301]. In FEUP InfoLab, we have developed the ANT (Ad hoc search of eNtities and Text) system[1], an academic search engine based on the information needs of the University of Porto. As ANT evolved into a viable prototype, there was the need for continued experimentation on a separate platform, while also providing features for debugging, performance measurement and documentation. Army ANT was born as an altruistic piece of software, built to preserve and improve ANT, but also to provide a workbench for innovation. It is based on the idea of freedom, to motivate creativity and novelty in information retrieval.

While Army ANT can be of general use for the evaluation of search approaches, it was also developed with some of the specific needs of entity-oriented search in mind. In particular, there are multiple tasks in entity-oriented search, such as ad hoc document retrieval (leveraging entities) [62, Ch.8], ad hoc entity retrieval [62, Ch.3], list search or related entity finding [62, Ch.4], that are not easily explored through conventional evaluation frameworks. Even if the unification of such approaches is a possibility, as we have previously proposed [302], this is not easily explorable with existing tools. Army ANT addresses this particular problem, providing, among other things, a way for different retrieval tasks to be explored and evaluated. For instance, the hypergraph-of-entity representation model, that we present in this thesis, supports multiple retrieval tasks, based on a unified ranking function, namely ad hoc document retrieval, ad hoc entity retrieval, or even term retrieval, where a ranking of related terms is provided for a given keyword query. This related term retrieval task was included merely as a possibility of the explored model, which is the kind of serendipity we aim to introduce through Army ANT.

In the remainder of this section, we describe Army ANT's system architecture, presenting the structure of collection readers, engines for the representation and retrieval of collections, databases for metadata storage, evaluators of effectiveness and efficiency, and the web server for searching, exploring, understanding and analyzing. Army ANT is currently available at FEUP InfoLab's GitHub[2], along with further documentation, under a BSD 3-Clause license.

#### 5.2.3.1 *Overview*

Army ANT is built with reusability, flexibility and cross-language interoperability in mind. In order to respect these requirements, we divided the system into what we consider the atomic components of information retrieval research:

1. Iterate over the documents in a collection (*reader*);
2. Index and search those documents (*engine*),
3. Eventually decorating them with metadata (*database*);
4. Assess the effectiveness and efficiency of the retrieval (*evaluator*);
5. Obtain as much additional information as possible about the system, in order to reiterate and improve (*web interface* ⇒ *learn mode*).

1 https://ant.fe.up.pt
2 https://github.com/feup-infolab/army-ant/

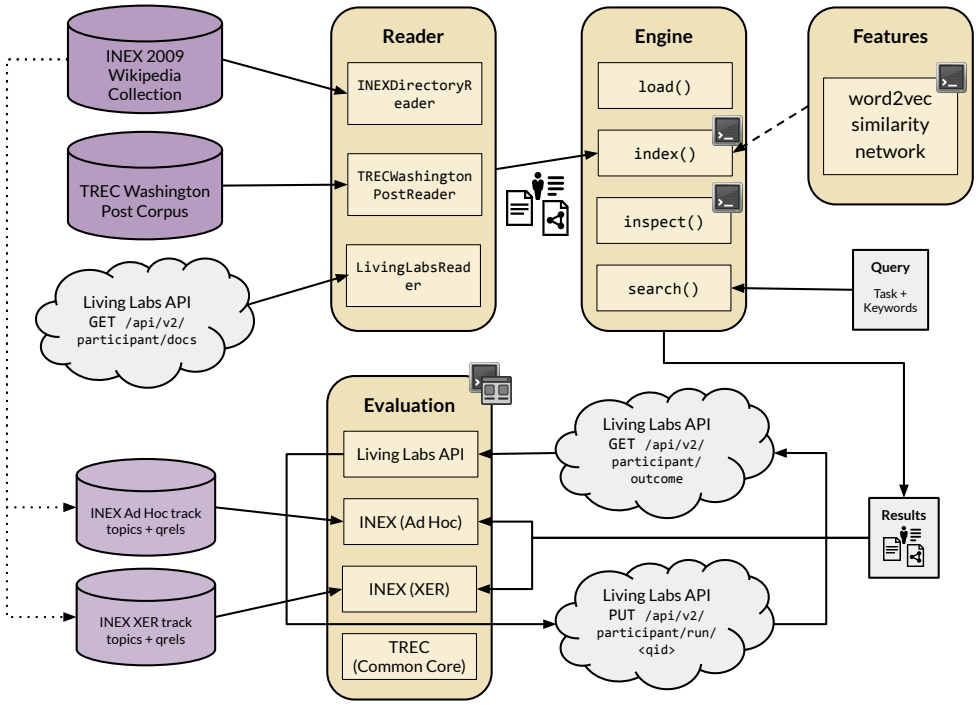



**Figure 5.11:** Army ANT system architecture.

Figure 5.11 provides an overview of the components in Army ANT, illustrating how they interact with test collections/APIs, as well as with each other. Solid arrows represent information flow, while dashed arrows represent optional interactions. Dotted arrows are simply used to indicate subcomponents of test collections (i.e., topics and qrels). In the figure, we can see some of the supported implementations, namely readers and evaluators for both both disk-based and REST-based data. We can also see that a query is defined as a task and a sequence of keywords, and that results can be based on documents, entities and their relations. Each component can have a command line icon, as well as a web interface icon, illustrating how it presents itself to the user.

Figure 5.12 provides an overview of the main dispatchable actions from the command line (`index`, `search` and `server`), along with the two main HTTP requests handled by the server (`GET /search` and `POST /evaluation`). It illustrates a typical workflow in Army ANT. Most actions, with the exception of the evaluation, involve contacting the engine to either index or search documents, as well as the database to either store or retrieve the metadata for a set of documents. Evaluation consists of queuing tasks that batch-retrieve results for a set of queries, which are then compared with a ground truth and assessed using metrics like the Mean Average Precision (MAP) or the required time to search.

#### 5.2.3.2 *Readers*

A process begins by iterating over a collection of documents and translating each document into a `Document` instance. A `Document` in Army ANT contains a unique document identifier (required), a text block (optional) and a knowledge block (optional). This is to reflect the focus on combined data and entity-oriented search, however any configuration can be considered with no impact for the implemented engines: only the text component; only the knowledge component (entities and/or triples); or both the text and knowledge components. Each document can also have a title, if one exists, as well as a dictionary of metadata that will be stored in a

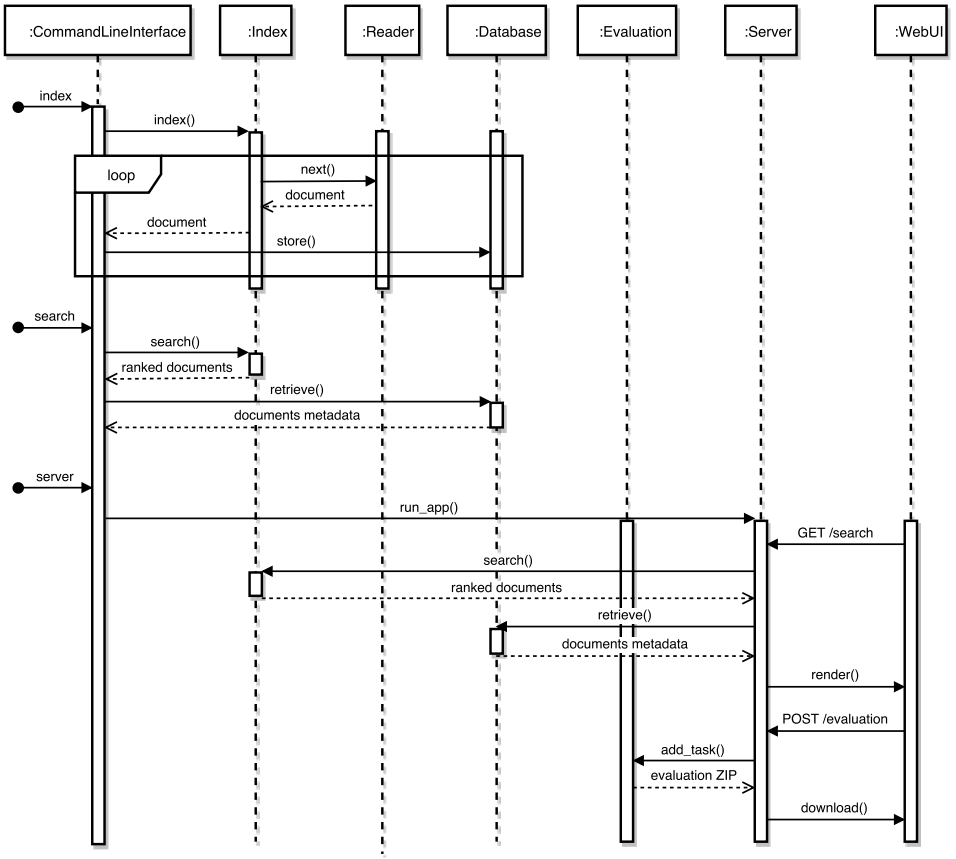



**Figure 5.12:** Sequence diagram for Army ANT.

`Database`. A reader is an iterator of `Document` and, in order to implement a reader, the following process must be followed:

1. Create a new class that inherits `Reader`;
2. Edit the `Reader.factory()` method to associate the new class to a name (used in the command line interface during indexing);
3. Implement the `__next__()` method, which should return a single instance of `Document`, or raise `StopIteration` when no more documents are available. This is where test collections are read and preprocessed.

We already provide six readers out-of-the-box, for five different collection formats: `CSVReader`, which uses column suffixes to identify the *doc_id* and columns representing values to append to the text block; `INEXReader` and `INEXDirectoryReader`, which can either load a single INEX 2009 Wikipedia collection [112] archive, or a set of archives within a directory, respectively; `WikipediaDataReader` to iterate over passages of the Wikipedia Relation Extraction Data v1.0 [267], `LivingLabsReader` , which reads documents directly from a Living Labs API endpoint [246], used for instance in TREC OpenSearch track [276], and `TRECWashingtonPostReader` for the TREC Washington Post Corpus[1], used for the first time in TREC 2018 Common Core[2] and News[3] tracks. Each reader uses a `source_path` parameter to pass corpus location information in the style of a JDBC connection string, as well as an optional `features_location` (e.g., document profiles) and `limit` parameters, whose usage is particular to each specific reader.

1 https://trec.nist.gov/data/wapost/
2 https://trec-core.github.io/2018/
3 http://trec-news.org/

#### 5.2.3.3 *Engines*

As previously stated, each engine is implemented through an `Index` subclass by defining its `index()` and `search()` methods. Army ANT provides two `Index` subclasses that serve as helpers or prototypes, in particular for two common cases: `ServiceIndex`, which simply splits the `index_location` into host, port and path components; and `JavaIndex`, which initializes the JVM along with common classes, through the `jpype` library. An `Index` can also implement an optional `load()` method, which is used to preload required components to memory. For example, we use the `load()` method in the `HypergraphOfEntity` to fully load the hypergraph, as well as auxiliary dictionaries, to RAM, in order to provide better performance, in particular for repeated web server search queries (singleton pattern). An engine can be implemented using the following steps:

1. Create a new class that inherits `Index`;
2. Edit the `Index.factory()` and `Index.open()` methods to instantiate the new class by name, for writing and for reading, respectively (used in the command line interface, for instance as part of the `index` and `search` commands);
3. Implement the `index()` and `search()` methods, which can be done one at a time, optionally taking advantage of the `Index.analyze()` method to tokenize text.
   - The `index()` method must yield instances of `Document` by iterating over a reader. While yielded documents are usually the same as the ones directly returned by the reader, in some cases multiple documents can be derived from each document provided by the reader (e.g., if the original document is split into multiple documents per section or passage, or based on its entities, to build virtual documents representing entity profiles [4]).

     There are also index extensions, which are a part of the index type and, by convention, should be appended to the base identifier using a colon and an extension identifier (e.g., *hgoe:syns:context*). Composite index types should then be dealt with in `Index.factory()` and `Index.open()`, and passed as a list to the respective engine.
   - The `search()` method must return a `ResultSet`, which is a list of `Result`. Each result must contain a *score*, an *id*, a *name*, a *type* and, optionally, any of the following fields:

     *metadata* usually containing the name and photo URL for the document (if the fields *name* and *img_url* are found, they are used in the web interface to display results);

     *components* an array of score components and values (e.g., for a particular document returned for a given query, the TF might be 2 and the IDF 1.3).

     The result set can also be associated with the following debugging information:

     *trace* a tree, where each node contains a message, describing a step in the ranking approach, and, optionally, a subtree of details (recursive definition).

     *trace_ascii* a string with a printable version of the trace tree (useful to include in experiment reports).

     The `search()` method can also take a task, a ranking function and its parameters as arguments. Supported tasks are defined in `Index.RetrievalTask`. Currently, three retrieval tasks are supported: `document_retrieval`, `entity_retrieval` and `term_retrieval`, but this can be extended as required.

We provide several baseline engines out-of-the-box: `LuceneEngine`, which supports TF-IDF, BM25 and divergence from randomness, respectively configurable as *tf_idf*, *bm25* with parameters *k1* and *b*, and *dfr* with parameters *BM* for the basic model, *AE* for the after-effect, and *N* for the normalization; and a `GraphOfWord` [16] implementation based on a collection graph [302]. The most recent release also includes `LuceneEntitiesEngine`, as a helper for building entity profiles, `LuceneFeaturesEngine`, as a helper for experimenting with query-independent features, and `TensorFlowRanking`, as an implementation of a learning-to-rank baseline that relies on the novel TF-Ranking library[1]. We also provide some of our own experimental implementations: `GraphOfEntity` [302]; and `HypergraphOfEntity` [303], a model using a hypergraph-based representation for combined data and a ranking approach based on random walks departing from seed nodes that are likely to represent the query.

#### 5.2.3.4 *Databases*

In Army ANT, we separate the actual index from the storage layer. We do this by chaining iterators: a reader yields documents that are indexed by an engine; an engine yields indexed documents, potentially with metadata, that are then read by a `Database` instance to handle metadata storage. This provides a flexible architecture, where a typical Lucene index, which can also act as a storage layer, can still be implemented to do so. This is done by dealing with storage in the engine layer directly and using the 'null' database iterator `Index.no_store`, which is automatically chosen when no database information is provided through one of the interfaces (cf. Section 5.2.4). On the other hand, if the indexing layer should be separated from the data storage layer, as is the case for `HypergraphOfEntity`, we can store metadata in a separate database.

Defining a database follows a similar scheme as the one used for the reader and engine. A `Database` subclass must implement the following two methods: `store()`, which iterates over an engine's `Document` instances, storing metadata for the provided *id*; and `retrieve()`, which extends a list of results (documents or entities) with their metadata, when it is not directly available from the index. Currently, we only provide a `MongoDatabase` implementation. We chose MongoDB since it was already a dependency for the evaluation module. The database module is secondary in Army ANT, being used only to ensure a user-friendly display of results when querying the interfaces.

#### 5.2.3.5 *Evaluators*

This module is based on the management of a job queue for evaluation tasks. Each task has a status indicator that can be in one of five states: *WAITING* (the task is queued, but inactive), *RUNNING* (results, usually for a set of given topics, are being retrieved and performance assessed), *DONE* (evaluation was successfully completed), *SUBMITTED* (used for online experiments, for instance via `LivingLabsEvaluator`), or *ERROR* (if it fails in a controlled manner).

Whenever a task is *DONE*, it is extended with a *results* field that contains a dictionary, where each key is a unique parameter identifier that points to a *ranking_params* dictionary, with further details, and to a *metrics* dictionary, containing effectiveness metrics and their respective values. A completed evaluation task can also be extended with a *stats* field, with a similar structure to the *results* field, but usually containing efficiency metrics, like the time per query for each topic (*query_time*) or the total and average query times (*total_query_time* and *avg_query_time*). Finally, each task also provides a downloadable zip archive containing intermediate results or additional details about the calculation of the evaluation metrics. The archive

1 TF-Ranking supports the learning of a groupwise scoring function [38], besides relying on the traditional pointwise, pairwise and listwise loss functions for training. It can be found at https://github.com/tensorflow/ranking.

might include a CSV with the average precisions per topic, as used to calculate the mean average precision, or multiple CSVs with the results for each individual topic, cross-referenced with the relevance per retrieved item, according to a ground truth. Next, we illustrate the file structure of an example archive created from a Lucene-based evaluation using TF-IDF.

```
5b4df784344f8640a29576dc.zip
├── eval_metrics.csv
├── eval_stats.csv
├── evaluation_details
│   └── no_params
│       ├── map_average_precision_per_topic.csv
│       ├── p_at_1000-precision_at_1000_per_topic.csv
│       ├── p_at_100-precision_at_100_per_topic.csv
│       ├── p_at_10-precision_at_10_per_topic.csv
│       └── precision_recall_per_topic.csv
└── search_results
    └── no_params
        ├── 2010003.csv
        ├── 2010014.csv
        ├── 2010023.csv
        ├── 2010032.csv
        ├── 2010038.csv
        ├── 2010040.csv
        ├── 2010049.csv
        ├── 2010057.csv
        ├── 2010079.csv
        └── 2010096.csv
```

Based on the tree, we can see that there are two summary files, *eval_metrics.csv* and *eval_stats.csv*, with the overall effectiveness and efficiency measurements. The *evaluation_details/* directory contains a subdirectory per parameter configuration, in this case *no_params/*, since TF-IDF does not take any parameter. Evaluation details include the average precisions per topic, as well as precisions at different cutoff values $n \in \{10, 100, 1000\}$ and the overall prevision and recall per topic, including true and false positives and negatives, as well as the F-measure for $\beta \in \{0.5, 1, 2\}$. Finally, the *search_results/* directory also contains a subdirectory per parameter configuration, each with a CSV per topic, containing the columns *rank*, *score*, *doc_id* and *relevant* (*"True"* or *"False"*, based on the ground truth provided in INEX 2010 Ad Hoc track). Besides the individual downloadable archive, with details about the particular task, there is also an overall summary table that combines the effectiveness metrics from each task. In order to simplify model comparison, this is previewable in the web interface and it can also be configured and downloaded as a CSV or LaTeX table (cf. Section 5.2.4.2).

In order to create an evaluator, it must be a subclass of `Evaluator` and implement a `run()` method to populate the *results* field with effectiveness metrics (e.g., MAP, NDCG@10) and the *stats* field with efficiency metrics (usually search times). We currently provide three evaluators out-of-the-box. The first is `INEXEvaluator`, which is based on the INEX 2009 Wikipedia collection and supports the topics and relevance judgments from the 2009 and 2010 Ad Hoc tracks, as well as the 2009 XML Entity Ranking track. A document is considered relevant whenever it contains a relevant passage (binary grading) and an entity is considered relevant for grade 1 and not-relevant for grade 0 (grade 2 judgments are converted to grade 0, considering 'not-an-entity' as not-relevant). The second evaluator we provide is for standard TREC topics and relevance judgments. `TRECEvaluator` was designed based on TREC 2018 Common Core track and, as such, it provides an additional TREC results file within the downloadable zip archive, which can be used with `trec_eval`. Finally, there is also a `LivingLabsEvaluator`, which submits a run for each query provided by a Living Labs API endpoint (e.g., TREC OpenSearch), assuming a previously indexed collection, usually based on the `LivingLabsReader` (cf. Section 5.2.3.2).

#### 5.2.3.6 *Secondary packages*

Secondary packages, by degree of importance, include: `analysis`, `features`, `sampling`, `extras` and `server`. These packages provide auxiliary features that complement or support experiments. More importantly, they provide a well-defined location for implementations that are commonly required for information retrieval research.

**ANALYSIS** The `analysis` package currently provides two functions for the retrieval and comparison of search results: `rank_correlation()`, to compare rankings from two different engines, and `rws_rank_concordance()` to measure the rank stability for a single engine based on the Random Walk Score (RWS). The `rank_correlation()` function receives parameters for two engines, including an index location and a type, as well as a ranking function and its parameters. The `rws_rank_concordance()` function receives parameters for an index location and type, as well as the length and number of iterations for the random walks in RWS. Both functions iterate over a set of topics from a file, and save the output to a given directory. Each function supports a *cutoff* parameter in order to obtain the 'Correlation@n' or the 'Concordance@n', respectively. A *method* parameter is also provided to select a different correlation or concordance metric (currently, only *"spearman"* and *"kendall"* are supported, respectively). Finally, the *repeats* parameter is also available to be used with nondeterministic ranking functions, such as RWS, repeating the experiment multiple times and providing an aggregated statistic for robustness.

**FEATURES** The `features` package includes subclasses of `FeatureExtractor`, which iterate over a collection using a reader and extract features from documents. A feature extractor must implement the `extract()` method. Features can then be processed by an engine, as an extension, during indexing, if the engine supports external features. At this time, only the `Word2VecSimilarityNetwork` is provided in the `features` package. It processes a corpus, extracting word embeddings and creating a similarity network, where each word is linked to the top-k words with a cosine similarity above a given threshold. The result is saved as a GZipped GraphML file that can be used when creating for instance a hypergraph-of-entity index with the *context* extension.

**SAMPLING** The `sampling` package was created as a generic way to implement independent collection samplers. We currently only provide an `INEXSampler`, which is based on a random selection of topics and the documents mentioned in the relevance judgments for those topics, optionally including linked documents. This is particularly useful either to produce a tiny development test collection, or simply to reduce the scale when building experimental retrieval approaches that are slower than conventional approaches.

**EXTRAS** The `extras` package contains additional features that are either specific to collections or indexes, and that do not fit any other package. We currently implement a function to fetch the first infobox image for Wikipedia collections that use the URL of the article as the *doc_id*. We also include a `word2vec_knn()` and a `word2vec_sim()` to explore the models generated by the `Word2VecSimilarityNetwork` feature. The first function retrieves the k-nearest neighbors of a given word, while the second function displays the cosine similarity between two words. Finally, we also provide a `fetch-dbpedia-entities` command that hasn't yet been refactored as a function in the `extras` package, despite being a part of the command line `extras` module.

**SERVER** The `server` package contains the RESTful services and the web interface. It is also prepared to receive documentation about supported collections and representation and retrieval models. We provide further details about the web interface in Section 5.2.4.2, and in Section 5.2.5.3 we illustrate how documentation can be created as a part of the `server` module.

### 5.2.4 Interfaces

In this section, we describe the command line interface and the web interface, illustrating how several of the features covered in Section 5.2.3 can be accessed. In particular, we show how indexing, search and evaluation can be done through the command line. We also describe how the web interface can be used to explore search results, illustrating several features of learn mode for visualizing and debugging the ranking functions, and for documenting collection and retrieval models.

#### 5.2.4.1 *Command line interface*

In this section, we describe the command line syntax, focusing on the most relevant packages (by pipeline order): `features`, `index`, `search`, `analysis` and `evaluation`. Each particular command can be run through `army-ant.py`. For example, for `search`, we would run:

```
./army-ant.py search <search-specific parameters...>
```

In order to list available parameters, for instance for `server` (which has default values for all parameters and can run parameterless), we can issue the following command:

```
./army-ant.py server -- --help
```

Some packages, such as `features`, `sampling` and `extras`, also provide subcommands, particularly for sets of tasks that are semantically similar, but do not share a common parameter set. For example:

```
./army-ant.py extras fetch-wikipedia-images <task-specific parameters...>
```

**FEATURES** During indexing, the data structure can be enriched with external features, through what we call index extensions (cf. Section 5.2.4.1). These features are extracted from documents, by iterating over a reader, and results are stored in the provided output directory. All features for a collection should be stored in a common directory, that will then be used by the `index` package as the base path to find the specific files or subdirectories that depend on the enabled index extension. Next, we show an example command for the extraction of a similarity network based on word embeddings:

```
./army-ant.py features \
  --method "word2vec_simnet" \
  --source-path "inex-2009-10t-nl/corpus" \
  --source-reader "inex_dir" \
  --output-location "/opt/army-ant/features/inex_10t_nl"
```

**INDEX** In order to create an index, we must decide which reader to use for iteration over the documents in a collection, and then select the index type and location. Optionally, we can also store document metadata in a database, which by default is MongoDB. This also requires the definition of a database name, otherwise resulting in metadata storage to be skipped. The metadata database can be shared by multiple indexes over the same collection and it's a step we are only required to run once for the first engine we are testing. Next, we show an example of an indexing run using the Living Labs API for the TREC 2017 OpenSearch track and

the graph-of-entity CSV representation (loadable to Neo4j and configurable as a `GraphOfEntity`):

```
./army-ant.py index \
  --source-path "http://api.trec-open-search.org::YOUR_API_KEY" \
  --source-reader "living_labs" \
  --index-location "/opt/army-ant/indexes/trec2017/graph-of-entity" \
  --index-type "goe_csv" \
  --db-name "aa_ssoar"
```

Other parameters, not displayed here, enable the definition of the database type and location, as well as a limit of documents to index, which is useful for testing purposes but might not be supported by all readers.

**Index Extensions** Representation models might have optional features, such as synonyms or context, which we call index extensions. Each individual extension can be toggled during indexing by a certain order, which, depending on the extension, might result in a different index data structure. An example of this is the hypergraph-of-entity with synonym and contextual similarity hyperedges:

```
./army-ant.py index \
  --source-path "localhost:27017/wapo" \
  --source-reader "wapo_dbpedia" \
  --index-location "/opt/army-ant/indexes/wapo/hgoe-dbpedia-syns-context" \
  --index-type "hgoe:syns:context"
  --features-location "/opt/army-ant/features/inex_10t_nl"
```

The previous command will read the TREC Washington Post Corpus from a *"wapo"* database on a *"localhost"* MongoDB instance and annotate it with DBpedia entities. The index will then be extended with synonyms based on WordNet [304] and contextual similarity based on the word similarity network extracted through the `features` command, as illustrated in Section 5.2.4.1. The colon is used as a separator for index extensions, added as a suffix to the index type, as show above. For this particular engine, specifying the index type as *"hgoe:syns:context"* is different from specifying it as *"hgoe:context:syns"*. In the first case, contextual similarity hyperedges will also exist between terms that were newly added from WordNet and did not previously exist in any document of the collection. In the second case, contextual similarity hyperedges will only consider terms that were already a part of the collection. In a different engine, such as `LuceneEngine`, index features might include, for instance, a payload weight based on the query-independent relevance of entities.

**SEARCH** Searching over an index requires a similar set of parameters to the one used during indexing. In particular, it requires information about the index and, optionally, about the database, if we want results to also display document metadata. It also requires a keyword query, as well as optional offset and limit parameters for pagination. For example, to search for `[ computer science ]` over the graph-of-entity for TREC 2017 OpenSearch track's SSOAR data, we run:

```
./army-ant.py search \
  --index-location "localhost:8182/goe_trec2017" \
  --index-type "goe" \
  --db-name "aa_ssoar" \
  --query "computer science"
```

**ANALYSIS** An analysis can be run over a collection, an index or even a set of extracted features. This means that each analysis has its own particular set of parameters. For instance, we might want to compare the rankings provided by two different ranking functions, through Spearman's coefficient of correlation $\rho$. Also, we might want to do this in a nondeterministic scenario (e.g., for a ranking function based on random walks). The following `rank-correlation` command does exactly this:

```
./army-ant.py analysis rank-correlation \
  --index-a-location "/opt/army-ant/indexes/inex-10t-nl/lucene" \
  --index-a-type "lucene" \
  --ranking-function-a "bm25" \
  --ranking-params-a "b=0.75,k1=1.2" \
  --index-b-location "/opt/army-ant/indexes/inex-10t-nl/hgoe-syns-context" \
  --index-b-type "hgoe:syns:context" \
  --ranking-function-b "random_walk" \
  --ranking-params-b "l=2,r=1000" \
  --topics-path "inex-2009-10t-nl/topics/2010-topics.xml" \
  --output-path "/opt/army-ant/analysis/inex_10t_nl-lucene_bm25-
      hgoe_syns_context-rank_correlation_at_10" \
  --cutoff 10 \
  --repeats 100
```

In this example, we compared two ranking functions for the same INEX collection. The first was Lucene BM25, with parameters $b = 0.75$ and $k_1 = 1.2$. The second was the hypergraph-of-entity's RWS (Random Walk Score), with parameters $\ell = 2$ and $r = 1{,}000$. For each ranking function and topic, the top-10 results were retrieved 100 times. Obviously this is redundant for BM25, which is deterministic, but not for RWS, which should converge, despite its nondeterministic nature. Spearman's $\rho$ is calculated for each topic, comparing the rankings provided by each ranking function. This is then repeated 100 times and averaged over all runs per topic. Both the averages per topic and the overall mean are saved to the output directory. Additionally, the Jaccard index, computed in the same manner, is also provided as a comparison metric of the documents retrieved by the two methodologies.

**EVALUATION** The recommended method to manage evaluation tasks is through the web interface, however we also provide a way to queue and run evaluation tasks from the command line. These tasks, along with results, will then be listed in the web interface. There is currently no way to access the results directly from the command line, but this is something that we will consider in the future. An example of a run for the *INEX (Ad Hoc)* evaluator, based on the INEX 2009 Wikipedia collection, along with topics and assessments from the 2010 Ad Hoc track, can be run as follows:

```
./army-ant.py evaluation \
  --index-location "/opt/army-ant/indexes/inex-10-nl/hgoe" \
  --index-type "hgoe" \
  --eval-format "inex" \
  --topics-filename "inex-2009-10t-nl/topics/2010-topics.xml" \
  --assessments-filename "inex-2009-10t-nl/assessments/inex2010.qrels" \
  --run-id "INEX - Hypergraph-of-Entity - Default Ranking"
```

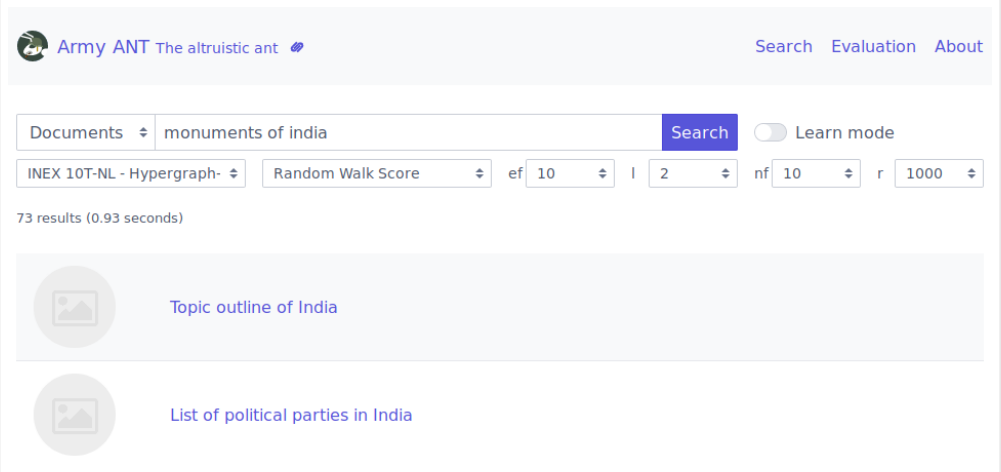



**Figure 5.13:** Basic search interface with engine selectors and learn mode toggle button.

#### 5.2.4.2 *Web interface*

The web server was developed using `aiohttp`[1], initially as a requirement to interact with `aiogremlin`[2], which is a Python library that can handle requests to the Apache Gremlin Server via the web sockets API. For the front end, we used an approach based on `jinja2`[3], which easily integrates with `aiohttp`, designing the interface using `spectre.css`[4]. Pages are rendered server-side, with some exceptions in the evaluation interface, where there is asynchronous task deletion, renaming, and restarting, implemented with plain JavaScript.

In the following paragraphs, we visually describe the components of the web interface. First, we cover the search component, which is closer to the expected user experience. We then describe the learn mode, for debugging and documenting. Finally, we describe the evaluation interface, where runs can be launched for different parameter configurations, in order to assess performance based on standard metrics.

**SEARCH** Figure 5.13 shows the basic search interface provided by Army ANT. It consists of a text box for the query (e.g., `[ monuments of india ]`), next to a combo box for the selection of the retrieval task (e.g., *"Documents"*, for ad hoc document retrieval). Below, we find two additional combo boxes, to select one of the configured engines (e.g., *"INEX 10T-NL - Hypergraph-of-Entity"*), as well as one of its ranking functions (e.g., *"Random Walk Score"*). Each ranking function can have a different set of parameters and available values, which are shown as separate combo boxes (e.g., *ef*, *l*, *nf* and *r*). Engine configuration is illustrated in Section 5.2.5.2. On the right, we can also find a button to toggle the learn mode. With the learn mode disabled, a search request will display a summary with the number of retrieved results and the time taken to run the query. It will also render each result either by displaying its *doc_id* or by using the *url*, *name* and *img_url* from metadata, if a database configuration is provided and these fields are available. The search mode also offers pagination and it's ideal for basic testing of ranking functions and their parameters.

**LEARN MODE** In order to better explore, understand and analyze the representation and retrieval model, Army ANT provides a learn mode. Depending on whether the developer of the selected engine has implemented the particular feature, learn mode can provide up to five diagnostic tools: a stripped down results list, show-

1 https://aiohttp.readthedocs.io
2 https://aiogremlin.readthedocs.io/en/latest/
3 http://jinja.pocoo.org/
4 https://picturepan2.github.io/spectre/

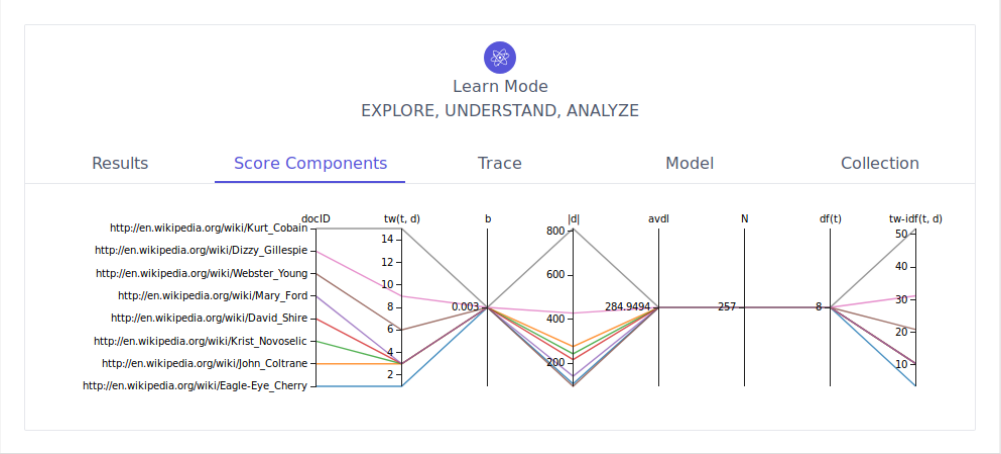



**Figure 5.14:** Learn mode: parallel coordinates visualization of the score components for a query to graph-of-word.

ing the *rank*, *score* and *id* for the top 30 documents; a parallel coordinates [305] score components visualization, to help understand the impact of each component in the final score and provide an intuition on whether components are correlated (Figure 5.14); a tracer, explaining in a sequential and hierarchical manner, and with increasing granularity, how the ranking function was implemented for the issued query over the given engine configurations (Figure 5.15); a textual description of the model, usually covering the representation model, the retrieval model and its ranking functions, the used *id*, and, when available, a citation to a research paper with further details; a textual description of the indexed collection, containing information about the source and temporal coverage of the data, as well as an example and, when available, a citation to a paper with further details.

**EVALUATION** The evaluation interface enables the measurement of performance for each engine, based on one of the implemented evaluators. Currently, four offline evaluators and one online evaluator are supported: *INEX (Ad Hoc)*, *INEX (XML Entity Ranking)*, *INEX (XML Entity Ranking - List Completion)*, *TREC (Common Core)*, and *Living Labs API*. The three INEX evaluators are based on the INEX 2009 Wikipedia collection, supporting topics and relevance judgments from the Ad Hoc track and the XML Entity Ranking (XER) track, respectively. These evaluators are useful to explore unified models for entity-oriented search, since the same test collection can be used to assess ad hoc document retrieval, ad hoc entity retrieval, and entity list completion. The *TREC (Common Core)* evaluator follows standard TREC formats and was implemented for experiments over the new TREC Washington Post Corpus, used both in the Common Core track and the News track, during the 2018 occurrence. It supports the preparation of runs to submit to TREC, which can be obtained through the zip archive associated with a completed evaluation task. Finally, *Living Labs API* provides an online method, based on team-draft interleaving [245], for evaluation. This was used in TREC OpenSearch over CiteSeerX (2016), the Social Science Open Access Repository (2016-2017) and Microsoft Academic Search (2016).

Figure 5.16 shows the evaluation task submission form. In order to queue a new evaluation task, you must first select one of the evaluators, as well as one of the engines, from the respective combo boxes. For the offline evaluators, you will at least have to provide a topics file, as well as an assessments file, provided by the respective evaluation forum. For online Living Labs evaluator, you will have to provide the base URL for the API, along with the API key. In any case, you are required to assign a run ID that will uniquely identify the evaluation task. The run ID can be renamed later and it will be used as the official run ID for TREC and Living Labs submissions.

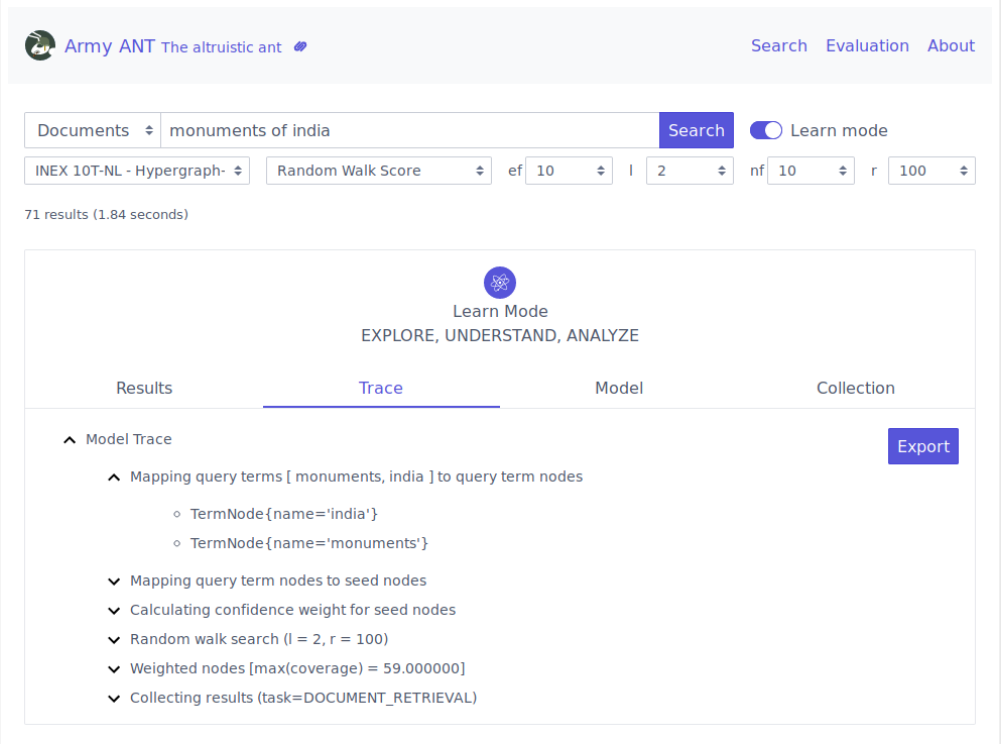



**Figure 5.15:** Learn mode: trace for random walk score in hypergraph-of-entity.

Each of the submitted tasks start in the *WAITING* status and can be selected by one of the active server threads — note that, by default, only one thread is launched by `aiohttp`, but multiple threads can be configured either through Systemd or Supervisor, as described in Section 5.2.5.1. Once a task is started, it will change to the *RUNNING* status and won't be available to other threads for processing. If the server is properly shutdown, through a *SIGINT* (or Ctrl+C), all *RUNNING* tasks will be reset to the *WAITING* status and restarted the next time the server is initialized. Once a task is completed, it will change to the *DONE* status, providing access to the evaluation metrics button [+]/[−], depicted in Figure 5.17, as well the downloadable zip button [⬇], to obtain further evaluation details. Any task in the *DONE* status will also be included in the download of the overall summary (Figure 5.18). There, you will be able to select a set of effectiveness metrics, the number of decimal places and other columns to include in the summary table, which can then be downloaded as a CSV, or a LaTeX file that you can include in your research paper. Since each researcher uses different performance metrics, we provide a configuration for the favorite metrics, so that researchers can quickly toggle the ones relevant to them.

### 5.2.5 Typical workflow

In this section, we present a typical workflow for developing and testing a new search engine using Army ANT. Through a use case, we describe the configuration process, as well as server deployment in a Linux system. We show how to setup either Supervisor or Systemd to launch multiple server threads or, as an alternative, how to use Docker Compose and automatically built Army ANT Docker images. Finally, we provide details on how to implement the components required to conveniently exploit learn mode over an engine, including how to implement a tracer, how to provide score component data and how to document a model.

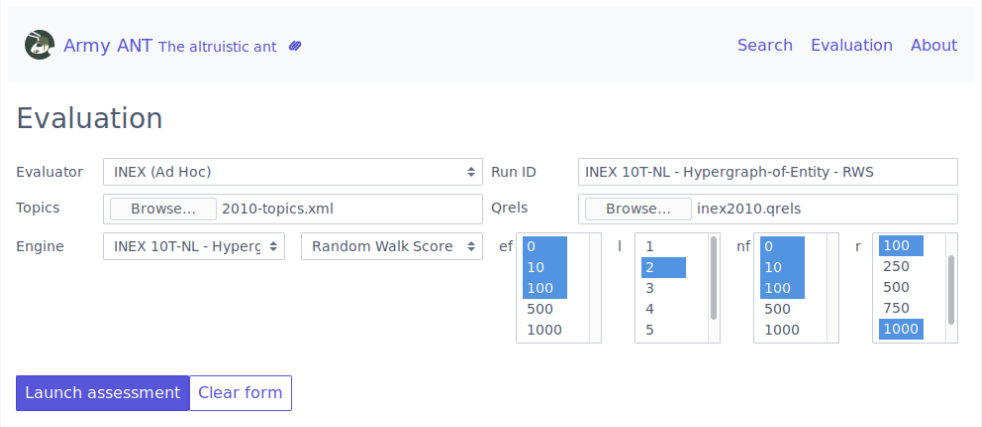



**Figure 5.16:** Evaluation task submission form.

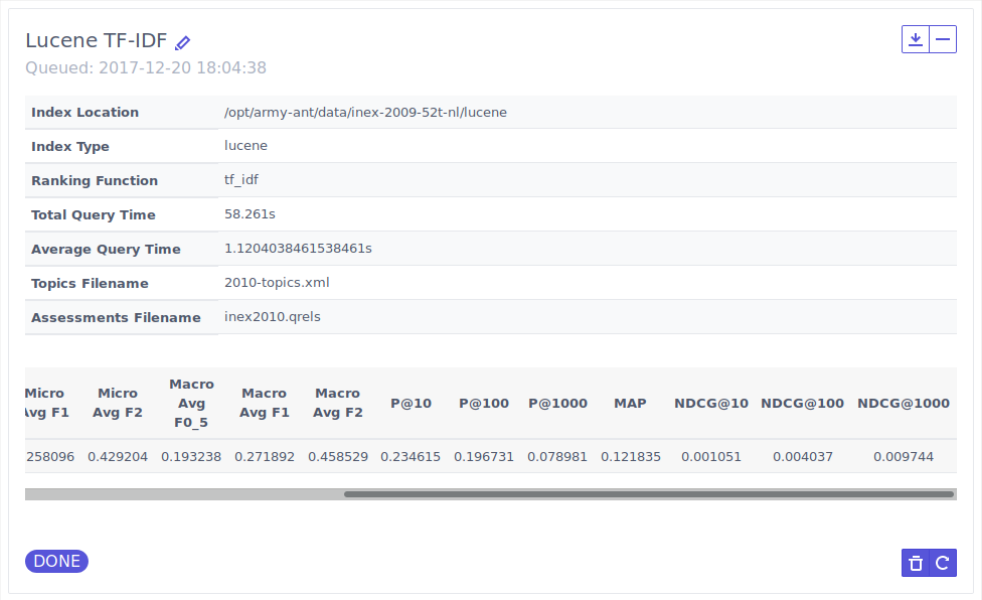



**Figure 5.17:** Evaluation task results for Lucene TF-IDF.

#### 5.2.5.1 *Use case*

Let us assume the following use case. We have an idea for a distributed index that we want to implement in Java, but, in order to do so, we would need a collection to index first. We already know that Army ANT provides an `INEXDirectoryReader` class that iterates over the individual documents of a set of archives for the INEX 2009 Wikipedia collection. We download the whole INEX collection to a *inex-2009/* directory, putting the archives in a *corpus/* subdirectory, the queries in a *topics/* subdirectory and the relevance judgments in an *assessments/* subdirectory. However, we would also like to have a smaller collection to use only during development. We can take advantage of the `sampling` command to do this:

```
./army-ant.py sampling inex \
 --qrels-input-path "inex-2009/assessments/inex2010.qrels" \
 --qrels-output-path "inex-2009-10t-nl/assessments/inex2010.qrels" \
 --topics-input-path "inex-2009/topics/2010-topics.xml" \
 --topics-output-path "inex-2009-10t-nl/topics/2010-topics.xml" \
 --corpus-input-path "inex-2009/corpus" \
 --corpus-output-path "inex-2009-10t-nl/corpus" \
 --query-sample-size 10
```

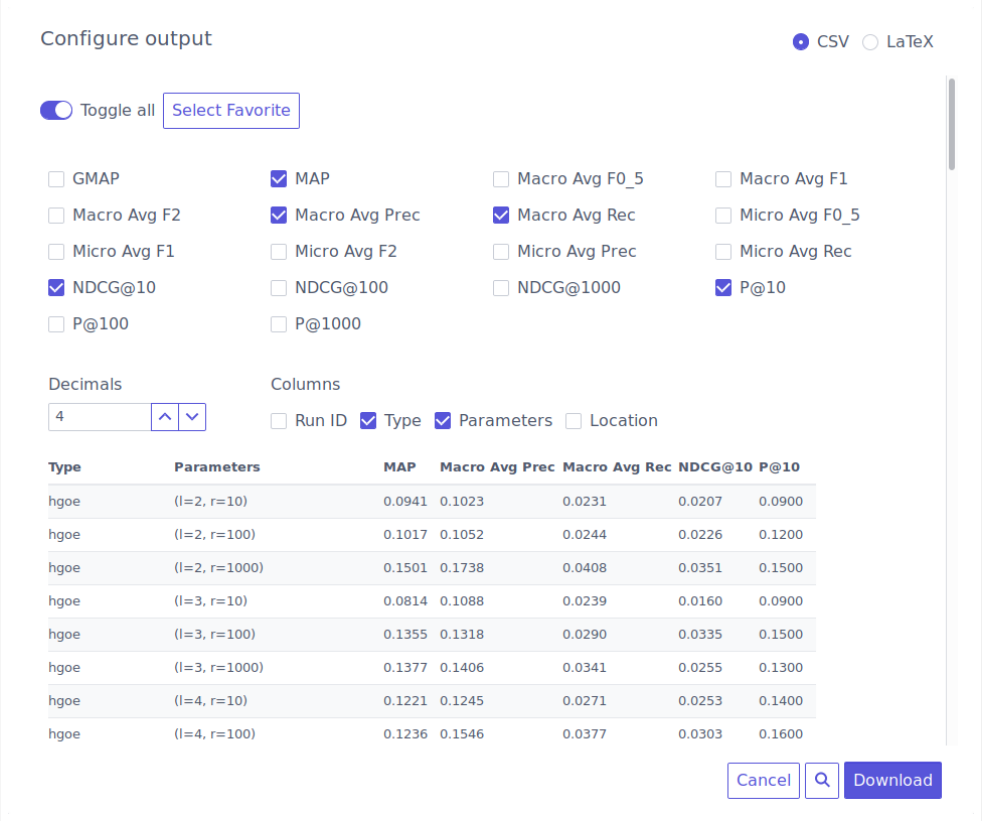



**Figure 5.18:** Evaluation session results export modal.

We can now create a new class within `index.py`, for instance named `DistributedIndex` that inherits from `JavaIndex`. We then open the *java-impl* project in the *external/* directory at the root of Army ANT and create a new subpackage under `armyant` for our implementation and we can optionally extend `Engine` to provide a basic structure to our index, as well as easy access to some utility methods such as the `analyze()` method. The Java implementation of Army ANT already provides a set of data structures, such as `Document` or `Triple`, that can be used in conjunction with `Engine` to streamline the process. We then go ahead and implement the `index()` method within a newly created `DistributedIndex` Java class and compile the project by running the following command from the root directory *java-impl*:

```
mvn compile assembly:single
```

Next, we can go to the Python's instance of `DistributedIndex` and implement `index()` simply by iterating over `self.reader` and creating instances of `JavaIndex.JDocument` from each iterated Python `Document`.

Before being able to index the INEX sample with `DistributedIndex`, we must first add an entry to the `Index.factory()` and `Index.open()` methods, providing a name for our engine (let's say we call it *distr*). We can now use the Army ANT command line interface to test the distributed index we created:

```
./army-ant.py index \
  --source-path "inex-2009-10t-nl/corpus" \
  --source-reader "inex_dir" \
  --index-location "indexes/distr" \
  --index-type "distr"
```

When everything works as expected with the indexing process, we can implement the `search()` method (cf. Section 5.2.3.3) in a similar manner and prepare for deployment.

#### 5.2.5.2 *Deployment*

In order to run the web server, we are required to edit *config.yaml*, setting up an engine by configuring parameters like index location and type, or supported ranking functions. We can then either launch a local instance directly using the *./army-ant.py server* command or deploy the server using Supervisor or Systemd for a more permanent and maintainable solution. For convenience, we also provide automatically built Docker images, that are installable through Docker Compose or available from Docker Hub[1].

**YAML CONFIGURATION** All Army ANT internal configurations are stored within *config.yaml*, which is divided in two main sections: `defaults` and `engines`. As shown in the following example, the `defaults` section contains: default database configurations (`db`), which can be overridden per engine; evaluation configurations (`eval`), including the favorite metrics used to configure the downloadable CSV or LaTeX comparison table (cf. Section 5.2.4.2); web service configurations (`service`); configurations for manually downloaded dependencies (`depend`); and JVM configurations (`jvm`), in particular for the heap size in mebibytes and other additional arguments.

```yaml
defaults:
  db:
    location: localhost
    name: army_ant
    type: mongo
  eval:
    metrics:
      favorite: [GMAP, MAP, NDCG@10, P@10]
    location: /opt/army-ant/eval
  service:
    ner:
      entity_list: /opt/army-ant/gazetteers/all.txt
  depend:
    stanford-ner: /opt/stanford-ner-2015-12-09
  jvm:
    memory: 5120
    other_args: -XX:+UseConcMarkSweepGC
```

In this example, the main database is *"army_ant"*, provided by a MongoDB localhost instance. This is used to store evaluation tasks, as well as document metadata. In this configuration, we also defined the favorite evaluation metrics to be the geometric mean average precision (GMAP), the mean average precision (MAP), the normalized discounted cumulative gain at a cutoff of 10 (NDCG@10) an the precision at a cutoff of 10 (P@10). The path for a list of entity names is also provided, as a setting for the named entity recognition service, while the path to Stanford NER is included as a dependency for NLTK 3.2.1 `StanfordNERTagger` wrapper, and the JVM is assigned 5 GiB of heap size and runs with the `ConcMarkSweepGC` garbage collector.

Let us now assume a configuration for the use case presented in Section 5.2.5.1. The goal is to add an entry with the configuration for the `DistributedIndex` engine. Let us also consider that it supports two ranking functions: TF-IDF, without parameters, and BM25, with parameters $k_1$ and $b$. The following example, shows a possible configuration for this use case.

1 https://hub.docker.com/repository/docker/jldevezas/army-ant/

```
engines:
  distr-inex-10t-nl:
    name: INEX 10T-NL - Distributed Index
    db:
      name: aa_inex
    index:
      type: distr
      location: /opt/army-ant/indexes/inex-10t-nl/distr
    ranking:
      default:
        id: bm25
      functions:
        tf_idf:
          name: TF-IDF
        bm25:
          name: BM25
          params:
            k1: [1, 1.2, 1.8]
            b: [0.5, 0.75, 1]
```

Each engine has its own unique key, usually descriptive of the type of index, but also the indexed collection (e.g., *"distr-inex-10t-nl"*, for a distributed index over the INEX 10T-NL collection). The engine should have a display *name*, as well as information about the index *type* and *location*. In the *ranking* section, we can define the default ranking function, out of a list of supported ranking functions that we provide. Each ranking function has its own display *name* and a dictionary of parameters and respective selectable values. Once the configuration is prepared, we can launch Army ANT's web server, either directly through the command line, usually for development and testing, or by deploying it through Supervisor or Systemd and an HTTP server such as nginx.

**SUPERVISOR/SYSTEMD AND NGINX** We provide example configuration files, in the *syadmin/etc* directory, for Supervisor (*supervisor/conf.d/army_ant.conf*), for Systemd (*systemd/system/army_ant@.service*) and for Nginx (*nginx/sites-available/army-ant*). Either install the Supervisor version or the Systemd version, by copying the configuration file to the corresponding system directory under */etc* and editing the paths to point to the Army ANT directory. For Supervisor, you can run `supervisorctl update` to activate the service, launching four server threads. For Systemd you must first activate the service by running `systemctl daemon-reload` and then run `systemctl start army_ant@{1..4}` to launch four server threads; you can also use `enable` in place of `start` to ensure the service is reactivated on system restart. Nginx will then expect four server threads as Unix sockets (the default), as well as a */etc/nginx/.htpasswd* that you can comment out if restricting access is not required. Simply copy the Nginx configuration file to */etc/nginx/sites-available* and then create a symbolic link in the *sites-enabled* directory, restarting Nginx.

**INSTALLING AS A CONTAINER VIA DOCKER COMPOSE** Army ANT provides automated Docker images for the git *master* branch, tagged as *latest* in Docker Hub, for the *develop* branch, tagged as *testing* in Docker Hub, and for non-release-candidate versions, tagged with the respective version (e.g., *0.4.2*). We also provide builds for specific tags, corresponding to releases used in events or mentioned in specific publications (e.g., *ecir-2020-demo*). Installation can be done by cloning Army ANT Install git repository[1] and running `docker-compose up`. Different docker containers can also be created per project, by running the `docker-compose -p <project-name> up` command instead. Two volumes are provided, the first for *config.yaml*, making it

1 https://github.com/feup-infolab/army-ant-install

editable in the local machine, and the second for the *data/* directory, used to transfer collections, indexes and other files between the local machine and the container (`/home/army-ant/data`). This means that, when indexing a new collection (e.g., *"inex-10-nl"*), we can simply copy it to the *data/* directory and run:

```
docker exec -i -t armyantinstall_army-ant_1 ./army-ant.py index \
  --source-path "/home/army-ant/data/collections/inex-10t-nl/corpus" \
  --source-reader "inex_dir" \
  --index-location "/home/army-ant/data/indexes/inex-10t-nl/lucene" \
  --index-type "lucene"
```

The created index will be easily accessible in the local machine through the *data/indexes/inex-10t-nl* directory.

#### 5.2.5.3 *Exploiting learn mode*

Army ANT provides a learn mode, for debugging and documenting the representation and retrieval models. However, in order for this feature to be useful, a new engine must first implement several components, including a tracer (best supported in Java), a dictionary of score components per retrieved document, and HTML+Jinja2 documentation for the collection and model — documentation can have dynamic elements, for instance to better illustrate active index extensions or the selected parameter values in the active query.

**IMPLEMENTING A TRACER** A trace is simply an ordered tree, where each node has an associated message and an ordered list of child nodes corresponding to details on the current node. The root node globally represents the retrieval model and nodes at progressively deeper levels will instantiate and illustrate retrieval details with an increasing granularity. The JSON serialization of a trace is illustrated next.

```
{
  "message": "Model Trace",
  "details": [
    {
      "message": "Mapping query terms [ rock, music ] to query term nodes",
      "details": [
        { "message": "TermNode{name='rock'}" },
        { "message": "TermNode{name='music'}" }
      ]
    },
    {
      "message": "Mapping query term nodes to seed nodes",
      "details": [
        { "message": "EntityNode{name='Music of Jharkhand'}" },
        { "message": "EntityNode{name='Doctor of Music'}" },
        { "message": "EntityNode{name='glam rock'}" },
        ...
      ]
    }
  ]
}
```

In the Java implementation, you can use the class `Trace` to help you prepare the JSON and ASCII serializations of the tree. It provides an `add()` method with syntax similar to `String.format()` to assign the message for the current node. It also provides `goDown()` and `goUp()` methods that can be called, for instance, before and after loops, as messages are sequentially added to the trace instance as new child nodes.

**Table 5.5:** Recommended metadata for collections and models.

| Field | Description |
|---|---|
| | Collections |
| Representation Model | Describe the data structure and the information encoded in the index. Index extensions should also be described here as a sublist with one item per extension. |
| Ranking Model | Explain the retrieval approach and weighting model. If there are multiple ranking functions, they can be described in a sublist, with one function per item. |
| Doc/Entity ID | Depending on the retrieval task, this could either be an identifier for a document, entity or another item. This field clarifies this and describes the format of the ID. |
| Paper | When possible, include a reference to the paper (or papers) that describe the model. Use a `<blockquote>` element for the bibliographic reference, with a `<cite>` element containing an anchor to the publication. |
| | Models |
| Source | Include an anchor to the online source of the dataset, usually providing a way to download or solicit the data. |
| Date | Temporal coverage of the data (i.e., period of time it refers to). |
| Description | Brief description of the data collection, including relevant information about the content, its format, origin, etc. |
| Example | Fragment of a document or item in the collection. |
| Paper | When possible, include a reference to the paper that describes the collection. Use a `<blockquote>` element for the bibliographic reference, with a `<cite>` element containing an anchor to the publication. |

**PROVIDING SCORE COMPONENTS** Score components provide a way to observe the values of individual components of the ranking function, instantiated for a particular document that was retrieved. Let us for instance consider a simplified weighting model based on TF-IDF, where the score of a document $d$ is given by the sum, over all query terms $t$, of the product between the term frequency in the document $\mathtt{tf}(t, d)$ and the inverse document frequency of the term $\mathtt{idf}(t)$. For each document, we could then discriminate between these two components for each query term. We illustrate this next with the *results* value of a JSON response from Army ANT's server, for a query with two terms.

```
[
  {
    "id": "1547719",
    "score": 3.45,
    "components": [
      { "docID": "1547719", "tf(t, d)": 3, "idf(t)": 0.9 },
      { "docID": "1547719", "tf(t, d)": 15, "idf(t)": 0.05 }
    ]
  },
  {
    "id": "17713850",
    "score": 2.5,
    "components": [
      { "docID": "17713850", "tf(t, d)": 3, "idf(t)": 0.5 },
      { "docID": "17713850", "tf(t, d)": 10, "idf(t)": 0.1 }
    ]
  },
  ...
]
```

Each object in a *components* array corresponds to one of the query terms. It contains keys and values for *docID*, *tf(t, d)* and *idf(t)*, which will be mapped to a dimension in the parallel coordinates visualization (Figure 5.14). It would have been an option to also add a dimension for the query term or for a document length normalization component, were they a part of the weighting model. Score components are a part of the `Result` classes in the Python, Java and C++ implementations.

**DOCUMENTING THE MODEL** Collection and model documentation is done directly in HTML+Jinja2, as part of the source code. In particular, each collection or model should have a corresponding file under *army_ant/server/templates/search/debug*, filed under *collections/* or *ranking_models/*, respectively. A documentation file should con-

**Table 5.6:** Dynamic variables available within documentation files.

| Variable | Description |
|---|---|
| *engine* | A string with the engine identifier (e.g., *"hgoe-inex-10t-nl"*). |
| *index_features* | An array of enabled index extensions (e.g., [*"context"*, *"syns"*]). |
| *task* | A string with the name of the retrieval task (e.g., *"document_retrieval"*). |
| *rankingFunction* | A string with a ranking function identifier from the active engine (e.g., *"random_walk"*). |
| *rankingParams* | A dictionary of ranking function parameters and their values as strings (e.g., {*l*: *"2"*, *r*: *"1000"*}). |
| *query* | A keyword query (e.g., `[ rock music ]`). |
| *debug* | A flag showing whether learn mode is enabled (*"on"*) or disabled (*"off"*). |
| *time* | The number of seconds required to rank and retrieve the documents (e.g., 4.23). |
| *offset* | Used to indicate where the current page begins. |
| *limit* | Maximum umber of results per page. |
| *numDocs* | Total number of retrieved documents. |
| *page* | Current page number. |
| *pages* | Total number of pages. |
| *results* | Array of result objects, containing an *id*, a *score* and a *components* array. |
| *metadata* | Document metadata, for the retrieved documents, as stored in the database and if available. |
| *trace* | Object with an ordered tree of messages and details. |
| *trace_ascii* | ASCII version of trace, to print in a command line or store in a research log. |

tain a `<dl>` element with `<dt>` and `<dd>` elements for the metadata recommended in Table 5.5. Each list entry can also make use any of the dynamic variables available in the context of the search and learn mode modules, as listed in Table 5.6. Documentation is then displayed based on the engine identifier, which should consist of a model identifier (e.g., *distr*), followed by a dash and a collection identifier (e.g., *inex*), with the same names as the respective documentation files. Remaining parts of the engine identifier string, after an optional second dash, will be ignored.

## SUMMARY

In this chapter, we presented the two main software contributions that we developed throughout this doctoral work: ANT, and Army ANT. For ANT, we focused on describing three main aspects: (i) search engine architecture; (ii) event ranking based on the entities and relations extracted from the news announcing the events; and (iii) query understanding based on query segmentation and semantic tagging, and supported on the entity catalog from ANT's knowledge base. For Army ANT, we focused on describing the system architecture, as well as the command line and web interfaces, illustrating a typical development and research workflow, with a focus on the implementation of new search engines, while taking advantage of learn mode to inspect the index and debug the ranking function. We also promoted the documentation of the representation and retrieval models, in a useful and dynamic way, so that other information retrieval scientists can benefit from our open source platform for innovating in this area, while exploring and evaluating their own work.

# Part III

# CONTRIBUTIONS

# 6 GRAPH-OF-ENTITY

## Contents



Managing large volumes of digital documents along with the information they contain, or are associated with, can be challenging. As information systems evolve towards intelligence, it increasingly makes sense to power retrieval through all available data, where every lead makes it easier to reach relevant documents or entities. Modern search is heavily powered by structured knowledge, but users still query using keywords or, at the very best, telegraphic natural language. As search becomes increasingly dependent on the integration of text and knowledge, novel approaches for a joint representation of corpora and knowledge bases present the opportunity to unlock new ranking strategies.

We tackle entity-oriented search using graph-based approaches for representation and retrieval. In particular, we begin by proposing a model called graph-of-entity, as a novel approach for indexing combined data, where terms, entities and their relations are jointly represented. We compare the graph-of-entity with the graph-of-word, a text-only model, verifying that, overall, it does not yet achieve a better performance, despite obtaining a higher precision. Our assessment was based on the INEX 2009 10T-NL subset, described in Section 4.1.1.2, which was created from a sample of 10 topics and respectively judged documents. The offline evaluation we do here is complemented by its counterpart from TREC 2017 OpenSearch track, where, during our participation, we assessed the graph-of-entity in an online setting, through team-draft interleaving.

The online evaluation at TREC 2017 OpenSearch was carried over the SSOAR site (Social Science Open Access Repository), using the *title* and the *abstract* as a text block and the remaining metadata as a knowledge block. Unfortunately, due to a technical problem with the OpenSearch track infrastructure, we were unable to obtain feedback for the real round during August 2017. As an alternative, we were offered the opportunity to participate in a third extraordinary round, happening during October 2017, as well as provided with available feedback from the period between the two official rounds, at the end of July 2017. We obtained an outcome of 0.375 for the graph-of-word and 0.167 for the graph-of-entity, based on only 29 impressions with clicks, out of a total of 4,683 impressions. According to this small number of clicked impressions, both models performed below the site's native search, with graph-of-entity performing below graph-of-word, in concordance with the offline evaluation.

The structure of this chapter is organized as follows:

- **Section 6.1** introduces several unification aspects that can be supported by graph-based models, both for the joint representation of corpora and knowledge bases, and for the generalization of retrieval tasks, along with preprocessing tasks, specific to entity-oriented search.
- **Section 6.2** presents the technologies that we used for these experiments, as well as a toy example, for describing our implementation of graph-of-word, an existing graph-based representation and retrieval model, as well as our own novel model for combined data, the graph-of-entity.
- **Section 6.3** describes the evaluation approach used to compare the graph-of-word and the graph-of-entity, based on a small subset of the INEX collection, as well as the participation in TREC 2017 OpenSearch, which relied on an online assessment approach based on team-draft interleaving.
- **Section 6.4** discusses the carried experiments and obtained results, after finding two underperforming retrieval models, one of them inconsistently so in regard to the literature. We justify the pursuit of further experiments to improve this graph-based model.

## 6.1 UNIFICATION OVER GRAPH-BASED MODELS

As the production of digital documents continues to increase, the answers we are looking for become harder to reach, particularly when relying only on identifiers and linked data to directly reach relevant content. Moreover, using a structured query language is frequently inappropriate for a regular user, who prefers natural language to express their information needs [306]. Full-text search is often the answer, but it inherently discards structure, which is extremely valuable to increase precision. In this work, we attempt to integrate unstructured text and structured knowledge in order to improve retrieval effectiveness in entity-oriented search tasks.

Search has evolved from keyword-based matching. Over time, it has grown increasingly dependent on semantic matching, largely supported on natural language understanding techniques. The need to integrate unstructured text and structured knowledge has substantially increased. In fact, one of the biggest challenges in semantic search is dealing with heterogeneity [147], in particular on the web, where a potentially unlimited number of topics exist. We tackle the problem of heterogeneity in entity-oriented search by proposing a unified graph-based model for terms and entities, where relations are seen as leads to be followed in the investigation of a given information need.

The more accurately a user's information need is identified through query understanding, and the better the information within a document is understood, the more likely the query will be matched with relevant documents or entities mentioned in those documents. This frequently results in improved retrieval effectiveness and, therefore, increased user satisfaction. What about when there is ambiguity? Can we always use entity linking to segment and semantically tag a query, discarding all other segmentations, even those which are equally likely? What if we were unable to provide an adequate answer to the users, even though the information they sought was available in the indexed corpus?

In the graph-of-entity, we integrate query entity linking into the ranking process, that is, a given entity in the graph is more relevant if it was reached from a nearby seed node (usually another entity) whose probability of being a good representation of the query is high (i.e., it has a high confidence weight). This probability models the certainty degree of the query entity linking process.

**Listing 6.1:** SPARQL query for the shortest path between *Axel A. Weber* and *Solingen* in DBpedia.

```
PREFIX : <http://dbpedia.org/resource/>
SELECT DISTINCT ?s ?o1 ?t
WHERE {
  VALUES ?s { :Axel_A._Weber }
  VALUES ?t { :Solingen }
  ?s [] ?o1 .
  ?o1 [] ?t
}
```

**Listing 6.2:** SPARQL query for the shortest path between *Axel Weber (athlete)* and *Solingen* in DBpedia.

```
PREFIX : <http://dbpedia.org/resource/>
SELECT DISTINCT ?s ?o1 ?o2 ?o3 ?o4 ?o5 ?t
WHERE {
  VALUES ?s { :Axel_Weber_(athlete) }
  VALUES ?t { :Solingen }
  ?s [] ?o1 .
  ?o1 [] ?o2 .
  ?o2 [] ?o3 .
  ?o3 [] ?o4 .
  ?o4 [] ?o5 .
  ?o5 [] ?t
}
```

For example, let us assume the ambiguous mention to *"Axel Weber"*, who, according to Wikipedia, can either be the athlete or the economist. Let us now assume that the query also mentions *"Solingen"*, which is the birthplace of *Jens Weidmann*, the successor of *Axel A. Weber*, the economist. Now, the probability of *"Axel Weber"* referring to the economist increases, but there might also be a longer path connecting *"Axel Weber"*, the athlete, to *Solingen*. We can easily check this using DBpedia's SPARQL endpoint, by manually testing increasingly longer paths between both *"Axel Weber"* individuals and *Solingen*. Listing 6.1 shows the SPARQL query for the shortest path between *Axel A. Weber* and *Solingen*, which are only linked by one other entity, *Jens Weidmann* — this is consistent with what we have already described. Listing 6.2 shows the SPARQL query for the shortest path between *Axel Weber (athlete)* and *Solingen*, which are linked by five other entities, through two distinct paths — no shorter path would link the two entities. While the query `[ axel weber solingen ]` is more likely to refer to *"Axel Weber"*, the economist, there might still be a niche where users could be searching for *"Axel Weber"*, the athlete, investigating whether there is a relation between the person and the location.

This type of unified approach is more prepared to take advantage of available information, discarding no lead, in order to provide the freedom to search for all matching items. We might say that word or entity disambiguation would happen "organically" during the process of ranking. The hypothesis is that this might improve effectiveness for search queries in the long tail [307], in particular by increasing recall without decreasing precision.

In the last few years, there has been work in graph-based approaches for information retrieval [15, 16], and also a growing need for unified models [85, 257, 258, 302]. While many solutions focus on the integration of signals obtained from text represented in an inverted index with signals obtained from external knowledge bases like Wikipedia [92], there have been few attempts at modeling text and knowledge in an unified manner, as a single data structure.

In this experiment, we build on the idea of the graph-of-word [16] to propose a novel graph-based model that combines text and knowledge within a single representation. The graph-of-word is a document-based graph [15] where terms are represented by nodes, providing directed links to the following $n = 3$ terms, as a way of capturing context. The graph-of-entity also captures the links between terms, still accounting for term dependence, but also the links between entities, similarly to RDF. RDF is frequently used to represent knowledge bases such as DBpedia (the structured version of Wikipedia), through semantic triples of subject, predicate and object, that can be seen as a graph and, perhaps more importantly, it also captures the links between terms and entities, in an attempt to connect unstructured and structured data. On one side, the model is capable of representing the properties of terms in a text document, as well as the properties of entities and relations in a

knowledge base. On the other side, it provides a way to cross reference all available information, independently of the source, as well as an opportunity to define a common set of operators that simultaneously work for text corpora and knowledge bases.

**TWO ASPECTS COMBINED** From the surveyed literature (Chapter 2), we can make two assumptions about entity-oriented search. First, structured data from knowledge bases, which is inherently representable as graphs, is a fundamental part of the semantic search process. Therefore, knowledge bases must somehow be integrated into the existing frameworks, which are mostly supported by inverted files. Many approaches exist to integrate signals from text and knowledge, but fewer common representation models have been proposed so far. Secondly, graphs have consistently been used to improve text retrieval, even outperforming weighting schemes such as BM25. Graphs can thus be used to represent text and are also frequently used to represent knowledge. It is definitely of value to study how to combine these types of graphs, in order to take advantage of the information locked within unstructured data through the integration of structured data — the knowledge base augments the text, through entities and their relations, and the text augments the knowledge base, providing leads to new information, seamlessly and through a common model (all are nodes in a graph).

What we propose is that the representation model for text and knowledge should be shared, using a graph data structure to capture discourse properties from text, relations between entities from knowledge bases, and term–entity associations based, at the very least, on potentially obvious relations between terms and entities (e.g., through substring matching). The ideal graph-based representation should: (i) capture information complexity, while avoiding redundancy; (ii) facilitate the cross-reference of information from distinct individual sources; (iii) propose a clear representation for combined data (text + knowledge) [92, Definition 2.3] that is easily extensible to other types of media. The open research question is whether or not such a combined data model will, through the unlocking of innovative weighting schemes, improve retrieval effectiveness. In this chapter, we propose and evaluate a baseline model, the graph-of-entity, which defines a graph-based representation for combined data, as well as a graph-based weighting scheme that can be used for entity ranking. We compare the graph-of-entity with an implementation of the graph-of-word, in order to position our baseline model within the state of the art.

## 6.2 REPRESENTATION AND RETRIEVAL

In our experimental workbench, we implemented the graph-based models using a graph database per index (Neo4j[1]) and the ranking functions using the Gremlin DSL[2]. The goal of this work was to propose a graph-based representation for combined data (text and knowledge), while using the graph-of-word as a text-only baseline. Figures 6.1 and 6.2 illustrate the graph-of-word and graph-of-entity models, described in the following sections, based on the first sentence of the Wikipedia article for 'Semantic Search' (i.e., our example collection consists of only one document with a single sentence):

1 https://neo4j.com/

2 Apache Gremlin is a domain-specific language for graph querying. More information at https://tinkerpop.apache.org/gremlin.html.

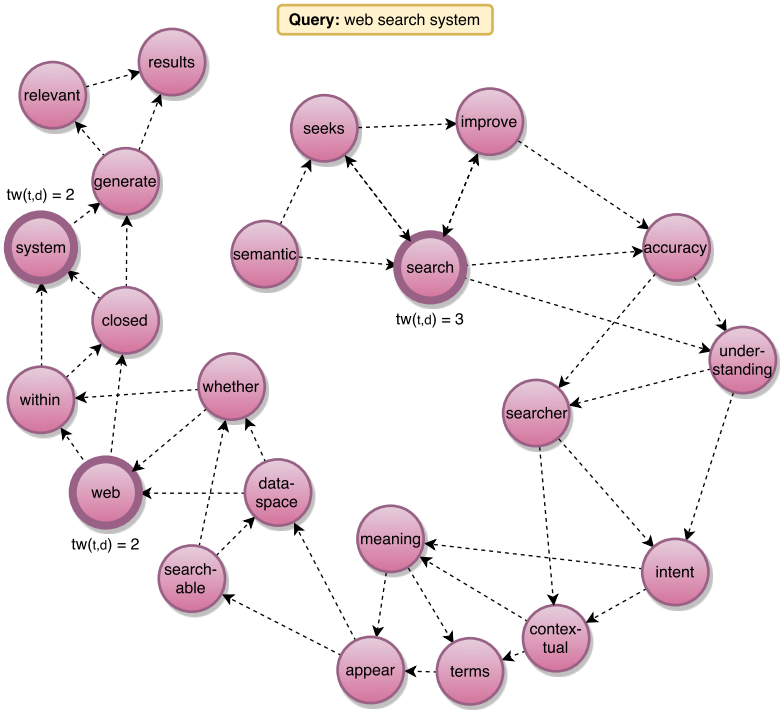



**Figure 6.1:** Graph-of-word (document-based graph; text-only) for the first sentence of Wikipedia's article on *Semantic search*. Nodes represent terms. Query term nodes are identified by a thicker border.

> *Semantic search seeks to improve search [Search Engine Technology] accuracy by understanding the searcher's intent [Intention] and the contextual [Contextual (language use)] meaning of terms as they appear in the searchable dataspace, whether on the Web [World Wide Web] or within a closed system, to generate more relevant results.*
>
> – Semantic search, Wikipedia, 09:10, 7 January 2016

### 6.2.1 Graph-of-word

**REPRESENTATION** The graph-of-word [16] is a document-based graph [15], where each node represents a term and each edge links to the following terms within a window of size $n$. The graph is unweighted, but directed, defying the term independence assumption of the bag-of-words approach. Figure 6.1 shows a graph-of-word instance for the first sentence of the Wikipedia article on 'Semantic Search', using a window size of $n = 3$. The graph-of-word is thus able to capture the context of each term within a particular document.

**RETRIEVAL** In the original graph-of-word implementation, the term weight (TW) metric was precomputed based on the indegree of each term node and stored in the inverted index to be used in place of the term frequency (TF). In our implementation, however, this was done in real time by filtering over the union of all document-based graphs and selecting a given subgraph based on a *doc_id* attribute stored in the edge. This is a less efficient solution, but it simplified the process of exploring and developing the novel graph-of-entity model, based on the graph-of-word, by defining a common representation framework. Additionally, the focus of our experiment was retrieval effectiveness; we were not particularly concerned with index efficiency.

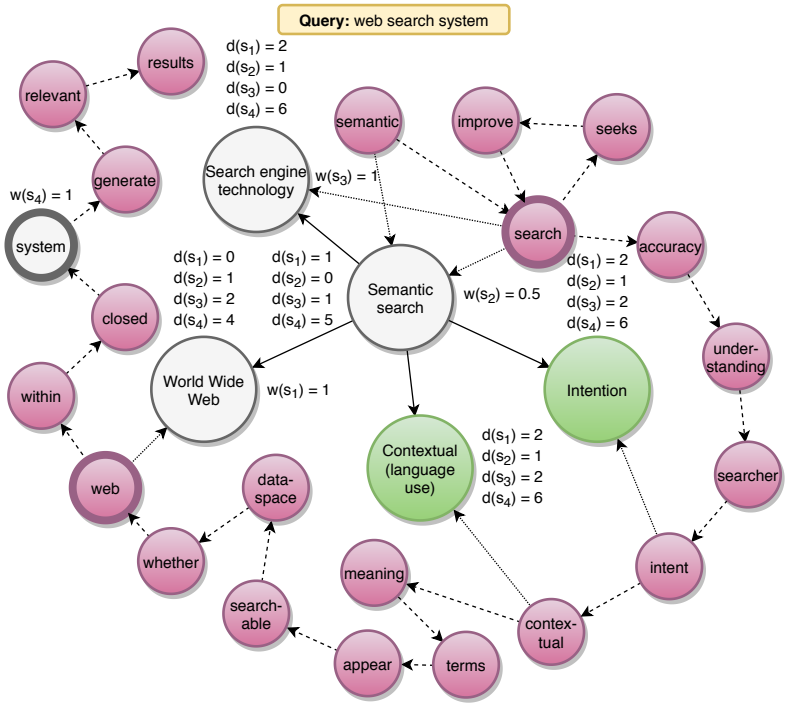



**Figure 6.2:** Graph-of-entity (collection-based graph; text+knowledge) for the first sentence of Wikipedia's article on *Semantic search*. Pink nodes represent terms and green nodes represent entities. A seed node for the given query is displayed in white. Query term nodes are identified by a thicker border.

Equation 6.1 shows the ranking function used for retrieval over the graph-of-word [16, Equation 7].

$$TW\text{-}IDF(t,d) = \frac{tw(t,d)}{1-b+b\times\frac{|d|}{avdl}} \times \log\frac{N+1}{df(t)} \tag{6.1}$$

The formula was derived from the TF-IDF approach as defined by Lv and Zhai [118], replacing the $tf(t,d)$ function by the $tw(t,d)$ given by the node indegree of term $t$, for document $d$, on the graph-of-word. For example, in Figure 6.1, we assume the query `[ web search system ]` and find that the largest term weight, $tw(t,d) = 3$, was assigned to 'search', while 'web' and 'system' were tied in second place with $tw(t,d) = 2$. The parameter $b$ was fixed at 0.003, since, according to the authors [16], it consistently produced good results across various collections, with $|d|$ representing the length of document $d$, $avdl$ the average length of all documents in the corpus, $N$ the number of documents in the corpus, and $df(t)$ the document frequency of term $t$ in the corpus. In our implementation, both $|d|$ and $avdl$ were approximated by the number of edges within the respective document-based graph, since we did all computations directly based on the graph.

### 6.2.2 Graph-of-entity

**REPRESENTATION** The graph-of-entity is a collection-based graph [15], where nodes can represent either terms or entities and edges can be of three types: *#term* –[ *:before* ]→ *#term*, *#entity* –[ *:related_to* ]→ *#entity* and *#term* –[ *:contained_in* ]→ *#entity*. While the graph-of-entity was inspired by the graph-of-word, it only captures term sequence instead of term context, through term→term relations, that is, the window size is always one. Additionally, we also encode entity→entity relations in the graph as a way of representing knowledge associated with the document (e.g., obtained from an information extraction pipeline applied to the text, or sim-

ply consisting of Wikipedia concepts linked in some manner). Finally, term→entity relations are established based on a substring matching approach, where a link between a term and an entity is created whenever the term is contained within the entity's name as a whole word (i.e., partial word matches are not considered). The goal for the first version of this model was to keep it simple (e.g., refraining from using similarity edges), but strongly connected (namely capturing all obvious relations). The main goal was to capture the properties of text, while modeling knowledge and establishing relations between text and knowledge.

**RETRIEVAL** We rank entities in the graph-of-entity based on the entity weight (EW) for an entity $e$ and a query $q$. A set of seed nodes $S_q$ are derived from query $q$, based on the links between query term nodes and entity nodes; when there are no entity nodes linked to a query term node, then the term node becomes its own seed node. This step provides a representation of the query in the graph, that will be used as the main input for the ranking function. Next, we present a formal definition for $EW(e, q)$, based on three main score components: coverage $c(e, S_q)$, confidence weight $w(s)$ for a seed node $s$, and the average weighted inverse length of the path between a seed node $s$ and an entity node $e$ to rank.

Let us assume a graph-of-entity represented by an attributed labeled multigraph $G_e$, similar to the one depicted in Figure 6.2, and a set of operations over $G_e$ to obtain a ranking of entity nodes with a *doc_id* attribute. Let $q$ be a query represented by a sequence of term nodes $q_n$ and let $e$ be an entity node that we want to rank (i.e., it has a *doc_id* attribute). Let $Sq$ be the set of seed nodes derived from query $q$. For each node $q_n$ that represents a term in query $q$, we obtain the set of seed entity nodes $S_{q_n}$ that are adjacent to term node $q_n$. Whenever $q_n$ has no entity node neighbors, $S_{q_n} = \{q_n\}$. The set $S_q$ of all seed nodes derived from query $q$ is then given by $S_q = \bigcup_{q_n} S_{q_n}$. This means that $S_q$ will contain all entity nodes adjacent to query term nodes, as well as query term nodes that are not adjacent to any entity node (i.e., they represent themselves). For example, in Figure 6.2, assuming query $q = q_1, q_2, q_3$, the seed nodes are given by $S_q = \{e_1, e_2, e_3, q_3\}$, where:

| Vertex | Name | Source | Vertex | Name | Source |
|---|---|---|---|---|---|
| | ENTITIES | | | TERMS | |
| $e_1$ | *Search engine technology* | $q_2$ | $q_1$ | *web* | – |
| $e_2$ | *Semantic search* | $q_2$ | $q_2$ | *search* | – |
| $e_3$ | *World Wide Web* | $q_1$ | $q_3$ | *system* | $q_3$ |

Let $p_{es}$ be a path between an entity node $e$ and a seed node $s$, as defined by a sequence of vertices $e, v_1, \cdots, v_{(\epsilon-1)}, s$ in the undirected version of $G_e$. Let $P_{es}$ be the set of all simple paths $p_{es}$ between $e$ and $s$. Assume the function $\epsilon(p_{uv})$ as the length of a given path $p_{uv}$ between vertices $u$ and $v$, representing the number of traversed edges[1].

Equation 6.2 can be read as the ratio between the number of paths linking entity node $e$ and seed nodes $s$ and the total number of seed nodes $S_q$. That is, the coverage represents the fraction of reachable seed nodes from a given entity.

$$c(e, S_q) = \frac{|\{s \in S_q | \exists p_{es} \in P_{es}\}|}{|S_q|} \tag{6.2}$$

1 In practice, we also defined a maximum distance threshold to compute the length of a path between two nodes. That is, no paths above the given threshold were considered. For this particular experiment, we used a maximum distance of one, which is an extremely conservative value.

Let $e_{ts}$ be the edge incident to both a term node $t$ and a seed node $s$. Equation 6.3 can be read as the confidence weight of seed node $s$. It represents the confidence that a seed node is a good representation of the query term it was derived from.

$$w(s) = \begin{cases} \frac{|\{e_{ts} \in E(G_e) | \forall t \exists q (t = q_n)|\}}{|\{e_{ts} \in E(G_e)\}|} & \text{if } s \text{ is an entity node} \\ 1 & \text{otherwise} \end{cases} \tag{6.3}$$

Finally, Equation 6.4 shows the ranking function for a given entity $e$ and query $q$.

$$EW(e, q) = c(e, S_q) \times \frac{1}{|S_q|} \sum_{s \in S_q} \left( \frac{1}{|P_{es}|} \sum_{p_{es} \in P_{es}} w(s) \frac{1}{\epsilon(p_{es})} \right) \tag{6.4}$$

The query is only used to obtain the seed nodes $S_q$ that best represent $q$ in the graph. This is analogous to a step in a query entity linking process. The remaining steps are quite straightforward. We obtain the average weighted inverse length of the path between each seed node $s$ and the entity $e$ that we want to rank. Assuming that the seed nodes are good representations of the query in the graph, the closer an entity is from all seed nodes, the more relevant it is — closeness is measured by the inverse length of the path. Given there is a degree of uncertainty associated with the selection of seed nodes, we scale this value based on the confidence weight of the seed node — an entity close to a high confidence seed node is more relevant than an entity close to a low confidence seed node, but an entity further apart from a high confidence seed node might be on par, or even more relevant.

## 6.3 EVALUATION

During the evaluation stage, we aimed at assessing the retrieval effectiveness of the graph-of-entity in comparison with a slightly altered implementation of the graph-of-word. Particularly, the document length $|d|$ and the average document length $avdl$, used for pivoted document length normalization [117], were calculated based on the number of term nodes per document, which appear only once per document — this means that we were only able to account for unique terms to obtain the document length in the graph-of-word. However, this change is not particularly critical, given the low sensitivity of the graph-of-word to document length [16, Section 5.3] (using $b = 0.003$ is close to using no pivoted document length normalization at all). That is to say, our implementation of the graph-of-word is only slightly different from the original and still provides a solid baseline.

### 6.3.1 INEX 2009 10T-NL

We prepared two indexes based on the 7,487 documents from the INEX 2009 10T-NL collection, one for the graph-of-word and another one for the graph-of-entity. For our experiment, each index was stored as a graph database. We then retrieved the results for each topic, labeling each entry using a binary relevance attribute based on whether there were any identified passages in the judgments file.

Table 6.1 shows the result of the assessment for this small subset of INEX 2009 Wikipedia collection. In particular, we present the precision for the first 10 results (P@10), the mean average precision for a maximum of 1,000 retrieved results (MAP), the normalized discounted cumulative gain for the first 10 results, using binary relevance grades (NDCG@10), and the overall precision and recall. As we can see, the Graph-of-Word (GoW) obtained the best overall scores, except for precision. Recall for the graph-of-word was nearly optimal (0.9816) and significantly above the

**Table 6.1:** Evaluation metrics for the graph-of-word (GoW) and graph-of-entity (GoE) based on INEX 2009 10T-NL (precisions and recall were [macro] averaged over all topics).

| Model | P@10 | MAP | NDCG@10 | Prec. | Recall |
|---|---|---|---|---|---|
| GoW | **0.3000** | **0.2333** | **0.3265** | 0.1085 | **0.9816** |
| GoE | 0.1500 | 0.0399 | 0.1480 | **0.1771** | 0.2233 |

**Table 6.2:** Average precision per topic for the graph-of-word (GoW) and graph-of-entity (GoE) based on INEX 2009 10T-NL. Highest and lowest average precision per model is shown in bold; results are ordered by decreasing average precision for GoW.

| Topic ID | Topic Title (Query) | Average Precision | |
|---|---|---|---|
| | | GoW | GoE |
| 2010038 | `[ dinosaur ]` | 0.6189 | 0.0069 |
| 2010057 | `[ Einstein Relativity theory ]` | 0.2899 | 0.1364 |
| 2010003 | `[ Monuments of India ]` | 0.2888 | 0.0000 |
| 2010079 | `[ famous chess endgames ]` | 0.2541 | 0.0448 |
| 2010023 | `[ retirement age ]` | 0.2513 | 0.0027 |
| 2010040 | `[ President of the United States ]` | 0.2408 | 0.0051 |
| 2010096 | `[ predictive analysis +logistic +regression model program application ]` | 0.2185 | 0.0410 |
| 2010049 | `[ European fruit trees ]` | 0.0756 | 0.0119 |
| 2010014 | `[ composer museum ]` | 0.0624 | 0.1185 |
| 2010032 | `[ japanese ballerina ]` | 0.0331 | 0.0315 |
| | **MAP** | 0.2333 | 0.0399 |

recall for the graph-of-entity (0.2233). Such a high recall also translated into a lower precision for the graph-of-word (0.1085), which was the only metric that was beat by the graph-of-entity (0.1771). This means that we were unable to improve Graph-of-Entity (GoE) over the baseline, as expected. Nevertheless, we obtained a better precision, which is encouraging, given our simplistic first attempt at designing a graph-based representation for combined data.

Given the small dimension of the dataset and in order to better understand the obtained MAP scores, in Table 6.2 we present the average precision for each topic. We also present the issued query and highlight the highest and lowest scores per model. As we can see, `[ dinosaur ]` achieved the highest average precision in graph-of-word, retrieving 703 results (425 relevant), but only 3 results (all relevant) for the graph-of-entity. The lowest average precision for the graph-of-word was achieved for `[ composer museum ]`, retrieving 1,674 results, out of which only 64 were relevant; this was beat by the graph-of-entity, retrieving 179 results, out of which 30 were relevant. The lowest average precision for the graph-of-entity was achieved for `[ Monuments of India ]`, retrieving only 2 results, none of which were relevant.

While the graph-of-entity clearly captures additional information, differing mainly on the lack of explicit representation of word context, overall it did not present an improvement over the graph-of-word. Our approach focused on assessing the effectiveness of the model, in order to iteratively improve it and eventually surpass existing state-of-the-art graph-based approaches through the integration of text and knowledge and using a collection-based approach. Despite the disregard for efficiency, at this stage, the complexity of the model and its inefficient implementation supported on a graph database were critical challenges in setting up an evaluation workbench with acceptable run times. While we did not index the full INEX 2009 Wikipedia collection, with over 2.6 million documents, we were able to index a smaller test collection, based on a sample of 10 topics and corresponding judged documents (INEX 2009 10T-NL), in order to obtain some insight. Additionally, during the participation in the TREC 2017 OpenSearch track [268] we had

been able to index the complete SSOAR[1] collection and evaluate the models in a real-world scenario, which acts as complementary information to the performance results we present here.

### 6.3.2 TREC 2017 OpenSearch track

During the participation in the TREC 2017 OpenSearch track, we were able to index the complete SSOAR collection and evaluate the models in a real-world scenario, thus complementing the performance results provided by the INEX experiment with additional information.

The evaluation in the OpenSearch track differs from classical TREC evaluation based on test collections. For this track, participants are provided with a Living Labs API, where they can register for a set of available sites. The API then provides documents to index and queries to generate rankings. Provided queries correspond to the most frequently issued within the site, thus increasing the chance they will appear again in the future. As one of the provided queries is issued by the site, a participant is selected and the results for participant and site are interleaved using team draft interleaving [245]. Evaluation is then carried based on the clickthrough rate and, for the assessment, we account for the fraction of wins of the participant over the site. In the 2017 edition, only the Social Science Open Access Repository (SSOAR) was available as a site, providing 39,492 documents to index, along with 1,165 queries (676 train queries and 489 test queries). See Section 4.1.3 for further details on SSOAR and the Living Labs API provided to participants.

Our goal was to compare the graph-of-word with the graph-of-entity, based on the fraction of wins either model obtained against the site's results. While the evaluation was carried individually for each model and compared with the site's search model, using a different set of queries, this provides initial feedback as to whether the graph-of-entity is comparable or performs better than the graph-of-word — the hypothesis is that by including structured data and providing a combined data representation approach the results will improve. We submitted and activated three runs, one during the trial round (*goe_trec2017*), and two during the real round (*gow_trec2017-real_round* and *goe_trec2017-real_round*), which we analyze in this section.

#### 6.3.2.1 *Technical issue*

Unfortunately, there was a technical problem with the load balancer on the side of the OpenSearch track infrastructure that resulted in our team receiving no feedback for the real round, during August 2017. The criterion for a given run from any participant to be selected is based on the lowest number of impressions it has received so far. A rather unpredictable issue with the priority strategy of the load balancer led to our runs never being selected. This happened for two main reasons: (i) the early activation of our run in July 17, 2017, which by itself would have posed no issues; and (ii) the fact that the SSOAR site was kept active throughout the two rounds, even when no round was scheduled to run. The combination of these two events resulted in a total of 4,000 impressions for our runs, during July 2017, that were never surpassed by any other participant during August 2017, possibly due to lower traffic during the summer. Since the runs from every other participant were continuously lower in number of impressions, our runs were never selected to be displayed and thus received no feedback during August 2017.

The organization of TREC 2017 OpenSearch Track acknowledged and detailed this issue and provided two options: (i) sharing the feedback from the end of July 2017, which is not available directly via the API, since there was no official round happening at the time; and (ii) run an extraordinary round during October 2017. While either option cannot be considered comparable with the approaches from other participants, our main focus was on comparing the two models we propose,

1 https://www.gesis.org/ssoar/home/

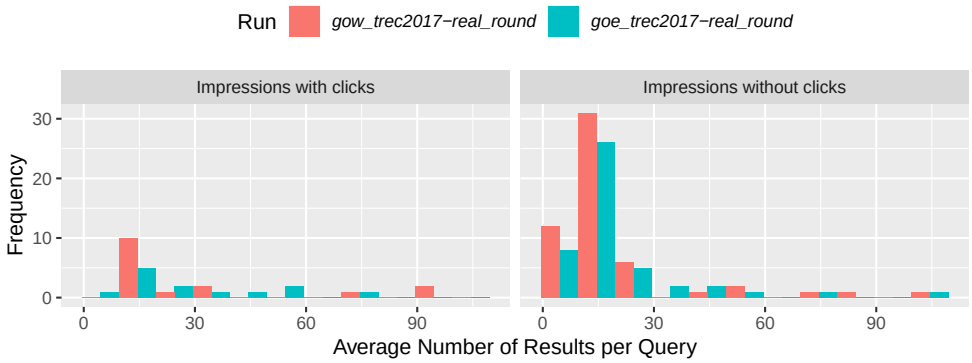



**Figure 6.3:** Result size distribution per run (bin width = 10).

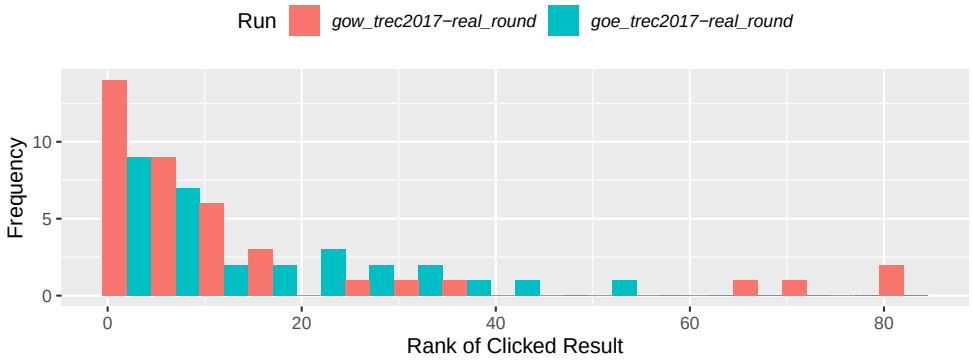



**Figure 6.4:** Rank distribution per run (bin width = 5).

making them both valuable. While the data from October 2017 was limited (only 97 impressions were provided, versus the 59 impressions from the failed official run), the data provided for the end of July 2017 was sufficient for a more in-depth analysis.

#### 6.3.2.2 *Dead period: feedback for July 17–31 2017*

Feedback collected between July 17, 2017 and July 31, 2017 corresponds to what we call a dead period, as no official round was scheduled to run at that time. This is not usually supplied to participants, however, given the technical issue described in Section 6.3.2.1, such data was provided to us as a JSON dump.

Figure 6.3 shows the average number of results per query, as provided to the users, based on the feedback for the dead period. Analyzed results were automatically generated by the Living Labs system from the interleaving of documents provided by the participant and the site. We distinguish between each run with different colors and separately analyze impressions with and without clicks. As we can see, the number of results varies between 0 and 100 and there is a significantly lower number of impressions with clicks. We also find that lists of results with less than 10 documents were never clicked, which might be an indicator of a poor precision at 10 for both models. Overall, the graph-of-word retrieves a slightly larger number of documents when compared to the graph-of-entity.

Figure 6.4 shows the rank distribution for clicked results. As we can see, most of the clicked results were at the top of the ranks and, overall, the graph-of-word achieved a higher number of clicks for the top ranks than the graph-of-entity.

Table 6.3 shows the outcome for the *gow_trec2017-real_round*, where we tested the graph-of-word (*gow*), and the *goe_trec2017-real_round*, where we tested the graph-of-entity (*goe*), based on feedback for the dead period. This enabled us to evaluate

**Table 6.3:** Outcome for the two graph-based models, during the dead period of July 17–31, 2017.

| Run | Impressions | | Outcome | Wins | Losses | Ties |
|---|---|---|---|---|---|---|
| | Total | Clicked | | | | |
| *gow* | 2,342 | 16 | 0.375 | 6 | 10 | 0 |
| *goe* | 2,341 | 13 | 0.167 | 2 | 10 | 1 |

the two graph-based models, by comparing each of them, individually, with the existing model used by SSOAR. This comparison was based on the outcome given by the fraction of wins for the participant (not including ties). For an outcome of 0.5, the two retrieval models would be equivalent, while for a value higher than 0.5, the participant's model would be better than the site's model. The results were not particularly encouraging, with both graph-based models achieving an outcome under 0.5 — 6 wins versus 10 losses for the graph-of-word and 2 wins versus 10 losses for the graph-of-entity. While there were over 2,300 impressions for each run, only a small fraction of about 15 impressions contained clicked results (~0.5%).

## 6.4 DISCUSSION

We tackled the problem of entity-oriented search through the proposal of a novel graph-based model for the representation and retrieval of combined data (text and knowledge). We proposed a collection-based representation of terms, entities and their relations (term–term, entity–entity, and term–entity), as a way to unify unstructured text and structured knowledge as a graph. We then proposed a very basic ranking function, supported on the graph-of-entity, where we mapped the terms of the query into nodes in the graph, preferentially expanding into neighboring entities, in order to obtain a query representation in the graph (seed nodes). We treated this as an open step in an entity linking process, that was only closed during ranking. Ranking was done based on the seed nodes, by treating them as leads. These leads were followed by trying to exhaust all available paths within a maximum distance, which resulted in the scoring of entity nodes. For evaluation purposes, not all entity nodes were ranked, limiting this operation to nodes that directly represented a document in the corpus (e.g., for Wikipedia, the entity mapped to the corresponding article, while, for SSOAR, a special entity had been created to represent the document). This enabled us to map the problem of entity ranking into the domain of documents, thus providing a way to evaluate using the traditional test collections and strategies that were available to us at the time.

The main goal of this work was to provide a simple baseline model that was graph-based and represented combined data in a unified manner. We performed evaluation based on a 10-topic sample of the INEX 2009 Wikipedia collection, which was complemented by the participation in TREC 2017 OpenSearch. During TREC, we were able to assess the two models, but only based on feedback from a "dead period" between the trial round and the real round, because of the technical issue described in Section 6.3.2.1. When comparing the graph-of-entity (our model) with the graph-of-word (a baseline text-only model), we found that, for the INEX experiment, our model did not outperform the baseline, except regarding precision. In TREC, both models also underperformed when compared to SSOAR's native search. Additionally, when comparing the models amongst themselves, we found no evidence of graph-of-entity performing better than the graph-of-word.

Results were unexpected, particularly for graph-of-word, as it had been compared to TF-IDF and BM25, outperforming both [16, Tab.2]. According to our research, in SSOAR, search is provided via DSpace, which uses a Lucene-based solution (either Elasticsearch or Apache Solr, depending on the version). We therefore assume

that the results provided by the graph-based models were being interleaved with results provided through a Lucene's *DefaultSimilarity*, usually based on TF-IDF, as implemented in *TFIDFSimilarity*. As such, the results we obtained are deemed inconclusive. If the graph-of-word had outperformed the site's search (i.e., TF-IDF) and the graph-of-entity had underperformed, we could have concluded that the graph-of-entity is a worse retrieval model. As it stands, however, we decided not to close this line of research and instead attempt to expand our evaluation to the complete INEX 2009 Wikipedia collection. With the graph-of-entity, we were nevertheless able to establish a graph-based strategy to jointly represent corpora and knowledge bases, as combined data, taking into account terms, entities and their relations in order to perform ranking. At the same time, we explored the integration of entity linking and entity ranking as a single task over the graph-of-entity, a first attempt at generalizing two different tasks.

## SUMMARY

In this chapter, we proposed and presented graph-of-entity, as the first attempt at a unified retrieval framework, where unstructured and structured data were jointly represented, in order to seamlessly contribute to the ranking process. We also explored the subsumption of named entity disambiguation by the process of ranking. We described our implementation of the graph-of-word baseline, a graph-based model developed by Rousseau and Vazirgiannis [16] to index text. We also described the implementation of the graph-of-entity, a model that builds on some of the ideas introduced in the graph-of-word, but was also expanded to include entities and relations from a knowledge base, representing the whole collection rather than a single document. Our goal was to provide a way to access all available information at any stage of retrieval. This meant the usage of atomic representations, including terms and entities, and their network of dependencies. Our goal was also to provide a way to harness the joint representation model to provide generalizations for information retrieval tasks. While we did not fulfill this goal in the current chapter, in the following chapters we build over this model to reach our goal of a unified framework for information retrieval, evaluated for entity-oriented search tasks. Meanwhile, in this chapter, we assessed this initial approach, based on a subset of the INEX 2009 Wikipedia collection, as well as by relying on the click-based implicit feedback provided from the online evaluation in TREC 2017 OpenSearch. We found that both the graph-of-word (our baseline) and the graph-of-entity (our model) were outperformed by the site's search algorithm (TF-IDF), which was inconsistent with previous literature [16], thus leaving this line of research open.

# 7 HYPERGRAPH-OF-ENTITY: FROM GRAPHS TO HYPERGRAPHS

## Contents



In the previous chapter, we have proposed the graph-of-entity as a purely graph-based representation and retrieval model, however this model would scale poorly. In this chapter, not only do we tackle this scalability issue by adapting the model so that it can be represented as a hypergraph, but we also describe several extensions that the index can consider, and a universal approach to ranking that can solve multiple entity-oriented search tasks. Relying on a hypergraph enables a significant reduction of the number of (hyper)edges, in regard to the number of nodes, while nearly capturing the same amount of information. Moreover, such a higher-order data structure, presents the ability to capture richer types of relations, including $n$-ary connections such as synonymy, or subsumption. We present the hypergraph-of-entity as the next step in the graph-of-entity model, where we explore a universal ranking approach based on biased random walks.

The structure of this chapter is organized as follows:

- **Section 7.1** reflects on the need for new representation and retrieval models able to better exploit all available information for solving information needs.
- **Section 7.2** describes the strengths and weaknesses of hypergraphs as instruments of generalization, commenting on performance differences between graphs, hypergraphs and fuzzy hypergraphs.
- **Section 7.3** delves deeper into the idea of a unified framework for information retrieval, describing which information can be expressed with a hypergraph, as well as proposing an approach for designing a universal ranking function for entity-oriented search tasks.
- **Section 7.4** presents the hypergraph-of-entity representation and retrieval model, introducing the base model, along with different configurations for *related_to* hyperedges, as well as four optional index features that can be combined as desired to extend the base model: synonyms, context, weights, and TF-bins (a novel concept that we propose). We show that, in this model, the data structure is the main agent of ranking, and describe a universal ranking approach based on seed node selection and random walks, covering its parameterization and the mapping of tasks to different input and output nodes and hyperedges.

## 7.1 CROSS-REFERENCING AND SOLVING GENERAL INFORMATION NEEDS

Information retrieval includes a wide range of tasks that frequently depend on, or at the very least benefit from, cross-referencing information locked within heterogeneous data sources. In entity-oriented search, there is frequently a combination of corpora and knowledge bases, and a strong reliance on the integration of unstructured and structured data. Perhaps the most straightforward approach to tackle this problem is to store text in an inverted index and entities and their relations in a triplestore, and then separately compute and combine signals from each storage unit [92, §5.1.2] [129, 308]. Despite the importance of the cross-referencing aspect, few attention has been given to the relations within and across corpora and knowledge bases in the design of representation models. Joint representation models have been explored through the usage of the inverted index, to store virtual documents built from passages mentioning the indexed entities [4, 131] and, while *mention* relations were implicitly captured in the process, no other relations (e.g., *entity* relations) are available in the model to inform retrieval. Representation learning has also been used to find a common word and entity embedding space [86] and, despite capturing latent relations between words and entities, these are not explicit or particularly exploited for retrieval. On the other hand, graph-based models are focused on the explicit representation of relations, be it intra-document, among terms [15, 16] or capturing syntactic and semantic dependencies [309, 310], inter-document, based on any type of links between documents [311], or even based on document-entity relations, resulting from the annotation of entity mentions that point to their instance in a knowledge base [312]. These are some of the reasons that make graphs viable to support a retrieval process based on the cross-referencing of information locked within text with information directly expressed as triples. Graphs are general data structures capable of capturing discourse properties from text [15], as well as knowledge from entities and their relations [50]. Graphs can support multiple tasks, from query understanding [313] to entity disambiguation [84] and document retrieval [16]. The challenge is to build such a model in a way that it is as complete and useful as possible, while remaining efficient.

In Section 1.3.1, we cited an example by Bast and Buchhold [74], showing that some relations can only be established through the connection of evidence atomically spread across corpora and knowledge bases in the form of terms, and entities and their relations, respectively. Figure 7.1 illustrates a case when fundamental information to answer the user's query is spread across corpora and knowledge bases. In particular, we can identify `[ astronauts who walked on the moon ]` completely from the knowledge graph. The same happens for `[ entertainers ]`. However, the friendship relation is only described in the text. Without a joint representation of such unstructured and structured data, and despite having access to the information, the search engine won't be able to correctly solve the user's information need. We agree with Bast and Buchhold, who argued that a new type of semantic index was required to solve this problem. While they presented a working solution, we propose that such a novel low level representation model would benefit from a graph-based approach and even more from a hypergraph-based approach. The real issue with the available indexing approaches is the lack of generalization. When a new type of resource needs to be indexed in the context of existing resources, the effort to change the representation model (i.e., the index) and the retrieval model (i.e., the ranking function) is usually considerable. With a graph-based model, however, there are multiple options to integrate resources, including semantic relations, similarities, co-occurrence, and so on. The obvious solution is to use information extraction to process the text and add the missing triple to the knowledge base: ⟨*Kevin Foster*, *:friendOf*, *Neil Armstrong*⟩. Despite the added computational complexity, this would be a solution for this specific problem of ad hoc entity retrieval.

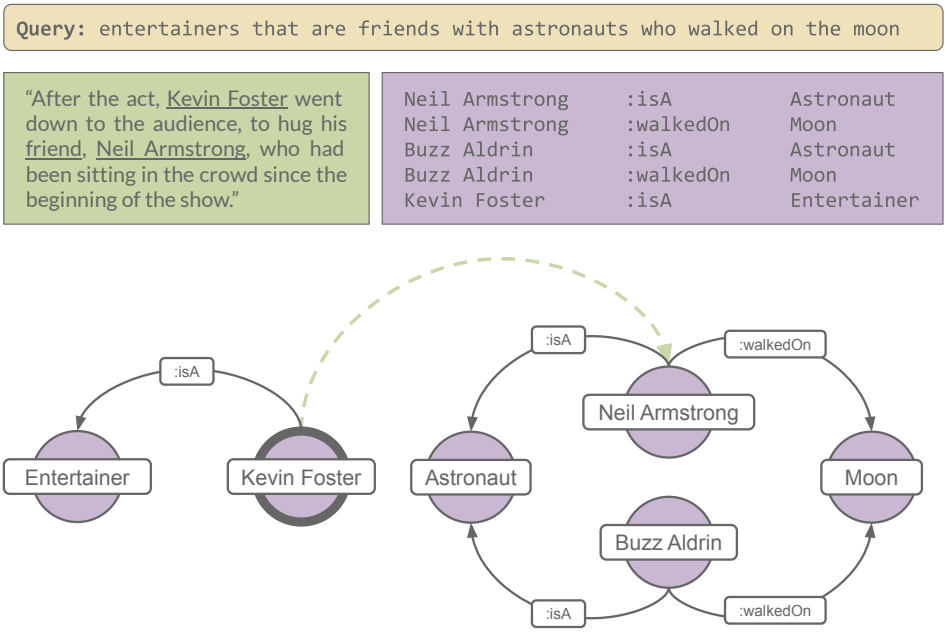



**Figure 7.1:** Cross-referencing information from corpora and knowledge bases. Green is associated with text, while purple is associated with entities. The dashed arrow illustrates the friendship relation inferred from the text.

Nevertheless, it would not fill the requirements for supporting other tasks, like ad hoc document retrieval, since the text would be transformed into structured data. In this work, we focus on general approaches to information retrieval, and in this chapter we shift our focus to hypergraphs.

## 7.2 HYPERGRAPHS: INSTRUMENTS OF GENERALIZATION

While we have frequently used data structures like graphs as a representation that promotes the integration of heterogeneous data, hypergraphs can take it even further. Not only they provide a more expressive data structure that can, at the same time, capture both the relations and the intersections of nodes, but they also reduce traversal complexity by grouping multiple nodes through hyperedges. Hypergraphs are a generalization of graphs where edges, or rather hyperedges, can contain an arbitrary number of nodes. Undirected hyperedges can be represented by a set of nodes (e.g., the terms in a sentence), while directed hyperedges can be represented by a tail set of nodes and a head set of nodes[1] (e.g., an e-mail to multiple recipients). Hypergraphs can be represented as an equivalent bipartite incidence graph, where each original node connects to a new node for each of its hyperedges. While such conversion is possible, it is not always ideal. This is true for instance for the study of overlapping or hierarchical relations. When broken down into multiple edges, they become harder to identify or read. Hypergraphs have been used to represent documents, but also to support ranking in information retrieval. Haentjens Dekker and Birnbaum [76] proposed a hypergraph-based representation of text as a hypergraph, and Bendersky and Croft [14] relied on hypergraphs to define a log-linear model based on higher-order term dependencies.

We propose that hypergraphs should be used as an alternative data structure for indexing, not only because of their expressiveness — a document might be mod-

1 The analogy is to an arrow, not to a list. This is why the tail set contains the source nodes and the (arrow)head set contains the target nodes.

eled as a hyperedge with terms and entities, potentially subsuming other relations between entities — but also because they have the potential to scale better than a graph-based approach — in particular, relations like synonymy or co-occurrence can be modeled with only one hyperedge as opposed to creating a complete subgraph for all synonyms or co-occurring nodes. It is also clearer, from a semantics perspective, to model synonymy or co-occurrence as a single hyperedge. Even visually, a hypergraph can, through transparency, provide further insights regarding intersections and subsumptions [314, Figures 2 and 5].

Let us for instance assume a labeled hyperedge, *related_to*, which connects four entities mentioned in a document entitled *"Cat"*: *Carnivora*, *Mammal*, *Felidae* and *Pet*. In subsumption theory, this hyperedge would represent an extension of *Carnivora*, *Mammal* and *Felidae*, a set of entities that could be a part of a hyperedge present for instance in a document entitled 'Lion'. While a lion is not a pet, it is still related to cat through the remaining three entities, so this information is useful for retrieval. While such example illustrates the potential of a hypergraph-based model, it only just scratches the surface. Using a hypergraph we can represent $n$-ary relations — linking more than two nodes (undirected hyperedge) or two sets of nodes (directed hyperedge) — but also hierarchical relations (hyperedges contained within other hyperedges) and any partial combination of the two. This means we can, for instance, model synonyms as an undirected hyperedge (e.g, $\{\mathtt{result}, \mathtt{consequence}, \mathtt{effect}, \mathtt{outcome}\}$) and even introduce hypernyms/hyponyms as directed hyperedges (e.g., directed hyperedge $\{\mathtt{cat}, \mathtt{lion}\} \rightarrow \{\mathtt{feline}\}$, or undirected hyperedges $\{\mathtt{cat}, \mathtt{lion}, \mathtt{feline}\}$ and $\{\mathtt{cat}, \mathtt{lion}\}$).

To sustain our argument, let us consider an alternative approach based on a tree, which is a type of directed graph, to represent hierarchical relations. With a basic tree we lose the ability to simultaneously represent hierarchical and $n$-ary relations. We argue that a bipartite graph or an edge-labeled mixed graph would allow for the representation of both $n$-ary and hierarchical relations, but, while conceptually it would contain the same information as the hypergraph, it would also be harder to read and use. We wouldn't be able, for instance, to naturally identify intersections or subsumptions. Independently of whether or not we translate the hypergraph to an equivalent graph, at the very least the theoretical modeling power of the hypergraph is clear. Nonetheless, in practice there are also some advantages to using hypergraphs over graphs, for instance the fact that a single hyperedge can store all synonyms at once, requiring a single step to retrieve the synonyms for a single term — $O(|V|)$ for term nodes $V$. Conversely, the same operation on a graph would require as many steps as the number of synonyms for a single term — $O(|V| + |E|) = O(b^d)$, assuming a breadth-first search approach for term nodes $V$ and synonym edges $E$ or, equivalently, for outdegree $b$ (the branching factor of the graph) and distance $d$ (where $d$ would be the same as the graph diameter).

Another advantage of hypergraphs includes the attempt to more closely model the human cognitive process. When we think, we inherently relate, generalize, particularize or overlap concepts. Most of us also translate natural language (sequences of terms) into concepts (entities), supporting the thought process on language. What we propose to do with the hypergraph-of-entity is to attempt to develop a kind of cognitive search engine (or at least the foundation for one). The way the (hyper)graph is traversed, including the selection of the point of origin, determines the kind of process over the "brain" of the engine. As a result, generalization becomes possible. With only slight adjustments to the search process, we can add support for multiple tasks from entity-oriented search. This includes ad hoc document and entity retrieval — the point of origin might be *term* nodes from a keyword query — as well as related entity finding and entity list completion — the point of origin might be one or several example *entity* nodes; both tasks can also be considered a type of recommendation [87]. In order to avoid a combinatorial explosion while still taking advantage of structural features, we propose that we model each process using random walks over the hypergraph — each step is based on the random se-

lection of an incident hyperedge and the subsequent random selection of one of its nodes. In this work, we focus on the task of ad hoc document retrieval, a process that we implement by modeling documents as hyperedges of terms and entities, and ranking *document* hyperedges through random walks. If we instead ranked *entity* nodes using the same strategy, we would have generalized the problem to ad hoc entity retrieval.

Claude Berge stated that hypergraphs could be used to simplify as well as to generalize [40]. We have shown that simplification can significantly reduce the order of complexity when compared to graph-based models like graph-of-entity. This is for instance the case with representing synonyms based on a single hyperedge rather than multiple edges in a graph. Moreover, it is not only a more natural, but also a more efficient modeling approach. On the other hand, the hypergraph is an adequate data structure to generalize, since it is able to represent clusters ($n$-ary groupings of elements), and other kinds of relations (e.g., directed relations between $n$-ary groupings of elements). It is also able to indirectly capture overlap through intersections, or hierarchies through subsumption, and it can factor uncertainty through node and hyperedge weights. During this doctoral work, however, we questioned the ability for generalization with hypergraphs, in particular when attempting to represent the frequency of a term (node) in a given document (hyperedge). Assigning a weight to the term would result in a global boost of the term in our collection-based hypergraph. A possible solution would be to define independent hypergraph models for each document. This would, however, segment information in a way that would inhibit the cross-referencing of atomic elements (i.e., terms and entities) at a low level, which is required for exploiting all available information, a condition that we established in Section 1.3. One solution that would rely on the collection-based instead of the document-based approach would be a fuzzy hypergraph [213]. In a (non-fuzzy) mixed hypergraph, undirected hyperedges can be represented as a set of nodes, while directed hyperedges can be represented as a tuple of two sets of nodes. However, in a fuzzy hypergraph, each set of nodes in the hyperedge contains tuples instead of nodes, and each tuple is a pair of node and membership weight. While this is not equivalent in space complexity to a graph-based representation where a pair of nodes would be connected by a weighted edge — three elements (two nodes and a weight), instead of two (a node and a weight) — it is a significantly more costly solution, not only regarding space complexity, but also regarding time complexity, since the ranking function would then be required to consider each weight. This makes the hypergraph a truly adequate solution for generalization, taking into consideration real-world restrictions, as opposed to only optimizing for the theoretical view. Relying on fuzzy hypergraphs is a solution that will result in a similar scaling behavior, when compared to the graph-of-entity that we explored in Chapter 6 and, while it is not impracticable, it is not ideal efficiency wise. Furthermore, as seen throughout this thesis, the hypergraph-based solution that we propose is, itself, not particularly efficient.

### 7.2.1 General concepts and definitions

We provide a mathematical framework, where we formalize several concepts and definitions, including relevant classes of hypergraphs, as well as useful properties and statistics, that we rely on across this manuscript.

#### 7.2.1.1 *Classes of hypergraphs*

In this section we formally define hypergraph, distinguishing between undirected, directed and mixed, as well as uniform and general.

**Definition 2** (Hypergraph). *Let $v$ be a vertex and $V$ be a set of vertices such that $v \in V$, with $n = |V|$ being the number of vertices. Let $E = E_U \cup E_D$ be the set of all hyperedges, where $E_U$ represents the subset of undirected hyperedges $e_U \in E_U$ and $E_D$ the subset*

*of directed hyperedges* $e_D \in E_D$*, with* $m = |E_U| + |E_D| = |E|$ *being the total number of hyperedges. Let also a set* $e_U \subseteq V$ *be an undirected hyperedge and a tuple of sets* $e_D = (t, h)$ *be a directed hyperedge formed by a tail set* $t \subseteq V$ *(source) and a head set* $h \subseteq V$ *(target). A hypergraph is then a tuple* $H = (V, E)$*.*

**Definition 3** (Undirected hypergraph)**.** *Under this notation, a hypergraph* $H = (V, E)$ *is said to be undirected when* $E = E_U$ *or, equivalently,* $E_D = \emptyset$*.*

**Definition 4** (Directed hypergraph)**.** *Similarly, a hypergraph* $H = (V, E)$ *is said to be directed when* $E = E_D$ *or, equivalently,* $E_U = \emptyset$*.*

**Definition 5** (Mixed hypergraph)**.** *It is also trivial to describe a mixed hypergraph* $H = (V, E)$ *when* $E_U \neq \emptyset \wedge E_D \neq \emptyset$*.*

**Definition 6** (k-uniform hypergraph)**.** *A* k*-uniform hypergraph is characterized by all of its hyperedges being defined over the same number* k *of vertices. For an undirected hyperedge* $e_U$ *it means* $|e_U| = k$*, while for a directed hyperedge* $e_D = (t, h)$ *it means* $|t| + |h| = k$*.*

* *Please refer to Banerjee and Char [315] for more information on directed uniform hypergraphs.*

**Definition 7** (General hypergraph)**.** *A general hypergraph is simply a non-uniform hypergraph, that is, with edges of diverse cardinalities.*

#### 7.2.1.2 *Hypergraph statistics*

In this section, we formally describe the hypergraph statistics that we rely upon for our analysis framework. In particular we describe the different degrees that can be computed for a vertex, the cardinalities of hyperedges, the diameter and average shortest path length, the clustering coefficient, and the density.

**Definition 8** (Vertex-based vertex degree)**.** *Let* $d_v(v)$ *be the degree of a vertex measured based on the number of adjacent vertices. Let* $E_v = E_{U_v} \cup E_{D_v}$ *be the set of incident hyperedges to* $v$*, ignoring direction,* $E_v^- = E_{U_v} \cup E_{D_v}^-$ *be the set of incoming hyperedges to* $v$*, and* $E_v^+ = E_{U_v} \cup E_{D_v}^+$ *be the set of outgoing hyperedges from* $v$*.*

*Vertex-based degree (ignoring direction) is given by:*

$$d_v(v) = \sum_{e_U \in E_{U_v}} |e_U| + \sum_{(t,h) \in E_{D_v}} (|t| + |h|)$$

*Vertex-based indegree is given by:*

$$d_v^-(v) = \sum_{e_U \in E_{U_v}} |e_U| + \sum_{(t,h) \in E_{D_v}^-} |t|$$

*And vertex-based outdegree is given by:*

$$d_v^+(v) = \sum_{e_U \in E_{U_v}} |e_U| + \sum_{(t,h) \in E_{D_v}^+} |h|$$

**Definition 9** (Hyperedge-based vertex degree)**.** *Let* $d_h(v)$ *be the degree of a vertex measured based on the number of incident hyperedges. Again, let* $E_v = E_{U_v} \cup E_{D_v}$ *be the set of incident hyperedges to* $v$*, ignoring direction,* $E_v^- = E_{U_v} \cup E_{D_v}^-$ *be the set of incoming hyperedges to* $v$*, and* $E_v^+ = E_{U_v} \cup E_{D_v}^+$ *be the set of outgoing hyperedges from* $v$*.*

*Hyperedge-based degree (ignoring direction) is given by:*

$$d_h(v) = |E_v|$$

*Hyperedge-based indegree is given by:*

$$d_h^-(v) = |E_v^-|$$

*And hyperedge-based outdegree is given by:*

$$d_h^+(v) = |E_v^+|$$

**Definition 10** (Hyperedge cardinality)**.** *Let* $c(e)$ *be the cardinality of a hyperedge measured based on the number of nodes it contains. Let* $e_U$ *be an undirected hyperedge and* $e_D = (t, h)$ *be a directed hyperedge.*

*Undirected hyperedge cardinality is given by:*

$$c(e_U) = |e_U|$$

*Directed hyperedge cardinality is given by:*

$$c(e_D) = |t| + |h|$$

In order to index hyperedges based on their number of nodes, we also use the notation $E_U^a$ to represent sets of undirected hyperedges of cardinality $a = |e_U|$, as well as $E_D^{a,b}$ to represent sets of directed hyperedges with a tail of size $a = |t|$ and a head of size $b = |h|$.

**Definition 11** (Diameter / avg. short. path len.)**.** *Let* $L = \{\ell_{u,v} : u \in e_U \wedge v \in e_U \vee u \in t \wedge v \in h\}$ *be the set of shortest path lengths between all pairs of nodes, where* $e_U \in E_U$ *and* $(t, h) \in E_D$*.*

*The diameter is then given by:*

$$\max L$$

*And the average shortest path length is given by:*

$$\frac{1}{|L|} \sum_{\ell_{i,j} \in L} \ell_{i,j}.$$

**Definition 12** (Clustering coefficient)**.** *The clustering coefficient measures the degree to which nodes tend to agglomerate in dense groups. We compute this metric based on the following approach by Gallagher and Goldberg [316]. Let* $E_v = E_{U_v} \cup E_{D_v}$ *be the set of incident hyperedges to* $v$*, ignoring direction. Let* $N(v)$ *be the set of all vertices adjacent to* $v$ *(i.e., sharing a hyperedge, while ignoring direction).*

*The clustering coefficient* $cc(u, v)$ *for a pair of nodes* $u$ *and* $v$ *is given by:*

$$cc(u, v) = \frac{|E_u \cap E_v|}{|E_u \cup E_v|}$$

*The clustering coefficient* $cc(v)$ *for a single node* $v$ *is given by:*

$$cc(v) = \frac{1}{|N(v)|} \sum_{u \in N(v)} cc(u, v)$$

*And the clustering coefficient* $cc(H)$ *for the hypergraph is given by:*

$$cc(H) = \frac{1}{|V|} \sum_{v \in V} cc(v)$$

**Definition 13** (Density)**.** *We transform a hypergraph* $H = (V, E)$ *into its corresponding bipartite graph* $G_H = (\mathcal{V}, \mathcal{E})$*, using the density of* $G_H$ *as an indicator of density for* $H$*.*

*The vertices* $\mathcal{V}$ *of* $G_H$ *are based on the vertices* $V$ *and hyperedges* $E$ *from* $H$ *and are given by:*

$$\mathcal{V} = V \cup \{v_e : e \in E\}$$

*The edges* $\mathcal{E} = \mathcal{E}_U \cup \mathcal{E}_D$ *of* $G_H$ *are established based on all pairs of vertices connected by a hyperedge* $E = E_U \cup E_D$ *from* $H$.

*The undirected edges* $\mathcal{E}_U$ *of* $G_H$ *are given by:*

$$\mathcal{E}_U = \{(u, v_e), (v_e, w) : e \in E_U \wedge u \in e \wedge w \in e\}$$

*And the directed edges* $\mathcal{E}_D$ *of* $G_H$ *are given by:*

$$\mathcal{E}_D = \{(u, v_e), (v_e, w) : e = (t, h) \in E_D \wedge u \in t \wedge w \in h\}$$

*Density* $D(H)$*, or simply* $D$*, is then given by:*

$$D = D(G_H) = \frac{2|\mathcal{E}_U| + |\mathcal{E}_D|}{2|\mathcal{V}|\,(|\mathcal{V}| - 1)}$$

## 7.3 A HYPERGRAPH-BASED UNIFIED FRAMEWORK FOR INFORMATION RETRIEVAL

In Section 1.4 we presented the problem of jointly representing corpora and knowledge bases, and proposing a universal ranking function to support multiple entity-oriented search tasks. In Chapter 6, we put forward graph-of-entity as a simple graph-based model to represent the relations between terms and entities. Retrieval relied on seed nodes that acted as a proxy for query representation. This allowed for the expansion of the original query keywords over the terms and entities of the graph. The second part of this expansion was similar to the candidate selection process for entity linking, but it delayed the ranking of the selected candidates, making it a part of the ranking process for the overall search strategy. This ensured that we were able to account for all available information, cross-referencing it to find the best results, while accepting uncertainty as part of the process. Ranking was done based on the distances to these seed nodes over all simple paths. The idea was that nodes closer to a higher number of seed nodes would have a higher probability of being more relevant. At the same time, disambiguating entities would occur "organically" based on the overall proximity to seed nodes.

Our initial effort had issues, regarding both the representation and the retrieval models. In particular, ranking based on all simple paths between seed nodes and all other nodes was inefficient, a process that was aggravated by the chosen document representation approach, which was based on edge labels — i.e., an edge originating from a document would be labeled according to the document's identifier, resulting in multiple edges for the same pairs of nodes, depending on the document (e.g., ten edges for $new \rightarrow york$ might exist if ten documents mention the city). Furthermore, as we have shown in Section 7.3, n-ary relations can be quite common and useful (e.g., synonyms), but costly to represent in a graph, increasing complexity and limiting experimentation even at a small scale. Moreover, combining multiple edges into a single hyperedge might lead to reduced complexity by diminishing the number of hyperedges in relation to the number of nodes. In this section, we continue to build the argument towards using hypergraphs as a joint representation model [§7.3.1] that can support a universal ranking function [§7.3.2], highlighting the strengths of this data structure and proposing a general approach to ranking.

### 7.3.1 Joint representation model

Graphs are a proven data structure for modeling heterogeneous data. They have been used to rank text documents [15, 16], as the underlying abstraction of hypertext [317–319], and for the representation of knowledge bases [47, 320]. Their ubiquity across the relevant areas of entity-oriented search led us to propose a graph-based representation and retrieval model that combines terms, entities, and their relations. As a reiteration of this approach, we now propose a model based on a weighted mixed hypergraph, since it can simultaneously and clearly express:

1. Undirected $n$-ary relations (e.g., bag-of-words, sentence, synonymy, context similarity);
2. Directed $n$-ary relations (e.g., a set of terms pointing to an entity, or a set of entities belonging to a category);
3. Hierarchical relations (e.g., subsumption);
4. Ontological relations (e.g., *Donald Duck* is both a *#duck* and a *#character* in a comic book);
5. Intersections (i.e., overlap is naturally captured by hyperedges and their shared nodes);
6. Uncertainty (e.g., knowing that there is 80% certainty that a set of terms are contextually similar can be translated into a weight in the respective *context* hyperedge);
7. User preference (e.g., a user rating 'Back to the Future' with 5 stars can be translated into a higher weight for the corresponding *entity* node).

Moreover, hypergraphs enable us to decrease the number of (hyper)edges in relation to the number of nodes, by prioritizing $n$-ary relations over binary relations. This is an advantage in reducing the complexity and improving retrieval efficiency.

In this thesis, we propose and study a hypergraph-based representation model, called hypergraph-of-entity, for the joint indexing of terms, entities and their relations. We explore most of the items that we previously listed, for expressing different relations in our model. In particular, we do not explore hierarchical relations, and we capture entity co-occurrence rather than explicit ontological relations, introducing uncertainty only on the weighted version of the model. This is further detailed in Section 7.4.1.

### 7.3.2 Universal ranking function

Regarding the retrieval model, we propose a universal ranking function over the hypergraph-of-entity. One of our ongoing goals is to design a function that can be used independently of the unit of information, as well as for multiple different tasks. We suggest this should be done by controlling:

- Input and output, e.g.:
  - Input *term* nodes to output a ranking of *document* hyperedges;
  - Input *entity* nodes to output a ranking of other *entity* nodes.
- Parameter configuration, e.g.:
  - Longer traversals are more exploratory;
  - Shorter traversals are more precise or on-topic.

In particular, the approach we propose is based on:

1. Finding a representation for the query in the hypergraph-of-entity;
2. Ranking nodes and hyperedges based on traversals around those representations;
3. Collecting only the relevant nodes or hyperedges to present to the user.

In this thesis, we propose a universal ranking function, which we evaluate for the tasks of ad hoc document retrieval, ad hoc entity retrieval, and entity list completion.

## 7.4 HYPERGRAPH-OF-ENTITY

Ad hoc document retrieval is traditionally a text retrieval task. Semantic search, however, frequently takes advantage of annotated collections, where entities are recognized and linked to external knowledge bases to improve document retrieval. In this work, we assume that, like entity annotations, relevant relations are also a part of the annotated document, extending it. Given a document containing a text block of unstructured data, as well as a knowledge block of structured information (i.e., entities and relations that are relevant to the document), our goal is to propose a joint representation model able to provide seamless integration, as well as support for entity-oriented search tasks, from ad hoc document retrieval to related entity finding. A regular document usually contains multiple text fields (e.g., *title*, *content*, etc.), which corresponds to the text block in the extended document. However, we also include a knowledge block, in the form of triples, that are usually available as structured data in the original document. The knowledge block can be directly extracted from a semi-structured document (e.g., building triples based on links to other documents), but it might also be obtained from an information extraction pipeline. There is no restriction about the source of the knowledge block, except that it should represent a set of triples related to the document. For example, the triples might represent co-occurring entities in a sentence or paragraph, or statements obtained from a dependency parser, or they could represent external knowledge about identified entities, from an external knowledge base.

We propose that a hypergraph would be the ideal data structure to represent a collection of extended documents, effectively capturing the dependencies and higher-order dependencies between terms and entities in relation to the documents. Take for example a *document* hyperedge created to associate all the elements within a document, including its terms and entities. Through higher-order dependencies we are, for instance, able to capture subsumption, where documents subsume (i.e., are more general than) relations between entities — we might interpret it as "document $d_1$ explains the relations between entities $e_1$, $e_2$ and $e_3$". The hypergraph-based model, detailed in Section 7.4.1, is able to capture multiple levels of information about the text, the entities and their relations, providing a more unified and insightful view over all available information. Although in this contribution we do not explicitly assess the impact of subsumption or hierarchical relations, but only of n-ary relations based on synonymy and context, we do highlight the ability for the model to capture such complex relations. Moreover, regarding ranking, document relevance scoring is based on biased random walks over the hypergraph, departing from a set of term and entity nodes that represent the query. This is a ranking approach that closely depends on the structure of the hypergraph, making it easier to track the impact that changes to the representation have in retrieval performance. The retrieval model is detailed in Section 7.4.2.

**Table 7.1:** Hypergraph-of-entity nodes and hyperedges (base model and extensions).

| Type | Set | Description | Observation |
|---|---|---|---|
| | | Nodes | |
| *term* | $V_t$ | Represents a single word from the original document. | In this work, the preprocessing pipeline includes: sentence segmentation; lower case filtering; replacement of URL, time, money and number expressions with a common placeholder, each; stemming via porter stemmer. |
| *entity* | $V_e$ | Represents an entity from the list of extracted entities and/or provided triples. | Each mention to an entity is modeled through this type of node (we consider disambiguation to be a part of the ranking). |
| | | Hyperedges (Base Model) | |
| *document* | $E_d$ | Represents a document through the set of all its terms and entities. | Undirected hyperedge. |
| *related_to* | $E_r$ | Represents a semantic relation between multiple entities. | Undirected hyperedge. Two implementations based on the triples from the collection: document co-occurrence; grouping by subject. |
| *contained_in* | $E_c$ | Represents a relation between a set of terms and an entity. | Directed hyperedge. In our implementation, this relation exists between terms that are a part of an entity name or mention and the corresponding entity node. |
| | | Hyperedges (Extensions) | |
| *synonym* | $E_s$ | Represents a relation of synonymy between a set of terms. | Undirected hyperedge. Present in the Synonyms model. The first synset from WordNet 3.0 is obtained for each noun term, missing terms are added to the model and the hyperedge is created. |
| *context* | $E_x$ | Represents a relation of contextual similarity between a set of terms. | Undirected hyperedge. Present in the Contextual similarity model. This is computed based on the top similar terms according to word2vec embeddings. |
| *tf_bin* | $E_b$ | Represents a sets of terms within the same term frequency interval, for a given document. | Undirected hyperedge. Present in the TF-bins model. The number of TF-bins per document is a parameter that can be set during indexing. |

### 7.4.1 An indexing structure for representation-driven retrieval

In this section, we introduce the variations of the hypergraph-of-entity representation model. This includes the base model, with two different configurations for *related_to hyperedges*, as well as multiple extensions based on synonyms, context, term frequency discretization, and weights.

#### 7.4.1.1 *An overview and formalization of the hypergraph-of-entity index structure*

The hypergraph-of-entity is a unified model for entity-oriented search. It provides a joint representation for corpora and knowledge bases, through a general mixed hypergraph (see Definitions 5 and 7). We propose several variations of this model that can be achieved through the extension of the base model, which is described in the sections that follow. As an introduction to the model, we present, in Table 7.1, an overview of all available types of nodes and hyperedges, while also providing the corresponding set for each particular element type. Based on this notation and supported on the concepts and definitions put forward in Section 7.2.1, we formally define the hypergraph-of-entity index structure.

**Definition 14** (Hypergraph-of-entity)**.** *A hypergraph-of-entity* $\mathcal{H} = (V, E)$ *is a general mixed hypergraph, where* $V = V_t \cup V_e$*, given the set of term vertices* $V_t$ *and the set of entity vertices* $V_t$*, and* $E = E_U \cup E_D$*, with the following* $E_U$ *undirected hyperedges and* $E_D$ *directed hyperedges:*

$E_U = E_d \cup E_r \cup E_s \cup E_x \cup E_b$*, such that each subset of undirected hyperedges represents a different type of structure:*

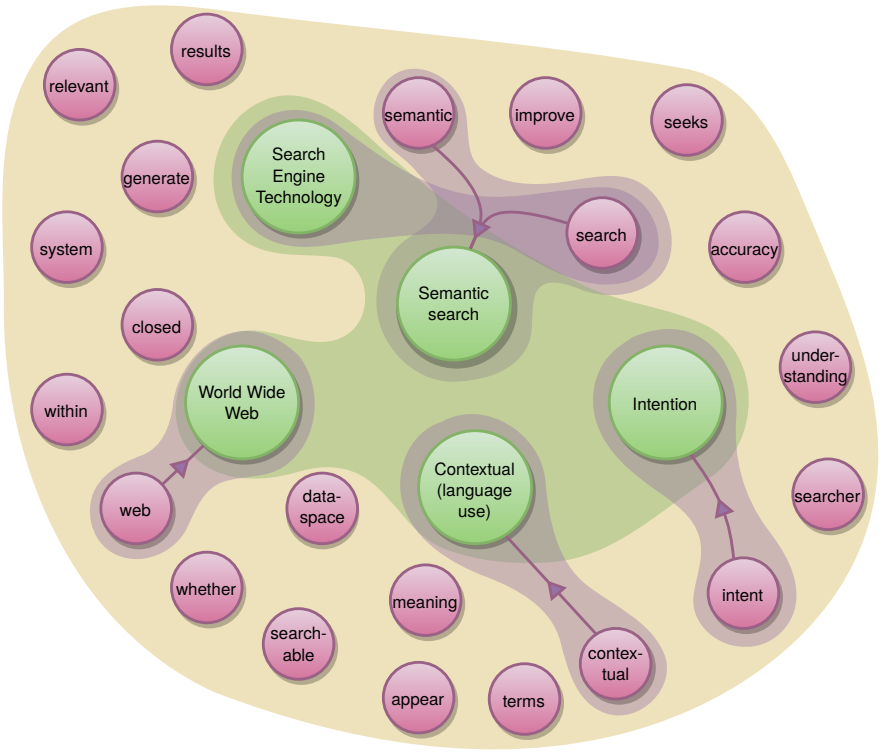



**Figure 7.2:** Hypergraph-of-entity base model, representing the first sentence of the Wikipedia article for *Semantic search*.

| | |
|---|---|
| *document* | $E_d = \{v : v \in V_t \vee v \in V_e\}$ |
| *entity relations* | $E_r = \{v : v \in V_e\}$ |
| *synonym terms* | $E_s = \{v : v \in V'_t \wedge V'_t \supseteq V_t\}$ |
| *contextually similar terms* | $E_x = \{v : v \in V''_t \wedge V''_t \supseteq V_t\}$ |
| *terms in the same TF range* | $E_b = \{v : v \in V_t \wedge v \in e_d \wedge e_d \in E_d\}$ |

*where* $V'_t$ *and* $V''_t$ *are extensions of* $V_t$ *(i.e., supersets), which may contain additional terms from external synonyms or contextually similar neighbors, respectively.*

$E_D = E_c$, *where* $E_c$ *is the only subset of directed hyperedges, such that* $E_c = \{(t, h) : t \subseteq V_t \wedge h \subset V_e \wedge |h| = 1\}$.

In the experiments we carry throughout this thesis, we control the structure of the index by enabling or disabling each of the described node and hyperedge types. For example, in Section 9.1, we explore the base model $\mathcal{H}_{bm}$, where $E_s = \emptyset$, $E_x = \emptyset$, and $E_b = \emptyset$. In the same section, we also explore several extended versions of the base model, namely by adding synonyms, which results in $|V_t(\mathcal{H}_{syns})| \geqslant |V_t(\mathcal{H}_{bm})|$ and $|E_s| > 0$. Another example can be found in Section 9.2, where we explore a text-only version of the model, such that $V_t \neq \emptyset$, and $V_e = \emptyset$.

#### 7.4.1.2 *Base model*

The hypergraph-of-entity is, in many ways, a simplification of the graph-of-entity (Chapter 6). In the graph-of-entity, the sequence of terms in a document was captured through term–term edges with a *doc_id* attribute, while in the hypergraph-of-entity we discarded term dependency in order to be able to model the terms within a document as a single hyperedge. This model is analogous to the bag-of-words, in the sense that term dependency is not captured by hyperedges (sets of nodes). Besides this major difference (one hyperedge per document), there are three other notable differences between the hypergraph-of-entity and the graph-of-entity: (i) each document hyperedge also contains nodes for entities mentioned within the document; (ii) sets of entities can be linked through a *related_to* hyperedge; and

(iii) sets of terms can be related to an entity through a *contained_in* hyperedge. We use a mixed hypergraph to represent a collection of documents. This means that hyperedges can be directed — from a set of terms to an entity (*contained_in*) — or undirected — sets of terms and entities (*document*), and sets of related entities (*related_to*).

In undirected hypergraphs, a set of nodes is a hyperedge. In directed hypergraphs, a hyperedge (or hyperarc) contains two sets of nodes — the set of source nodes is called tail, while the set of target nodes is called head. In the hypergraph-of-entity, we always have tail sets with cardinality one (for directed *contained_in* hyperedges) — this characteristic might be useful for defining a tensor representation of the hypergraph. Figure 7.2 provides a basic illustration of this model, capturing hyperedge direction, when it exits, through a multiple source and target single arrow. In the figure, pink nodes represent terms and green nodes represent entities. All term and entity nodes are linked by a yellow undirected hyperedge that represents the document as the set of its terms and entities. Entity nodes are linked by green undirected hyperedges, when the entities are related (e.g., through a property in an ontology). Sets of term nodes are linked to an entity by a pink directed hyperedge, whenever the terms are a good representation of the entity (e.g., through substring matching).

In particular, Figure 7.2 represents a single document based on the first sentence of the *Semantic search* Wikipedia article. This was already introduced in Section 6.2, but we replicate it here for convenience:

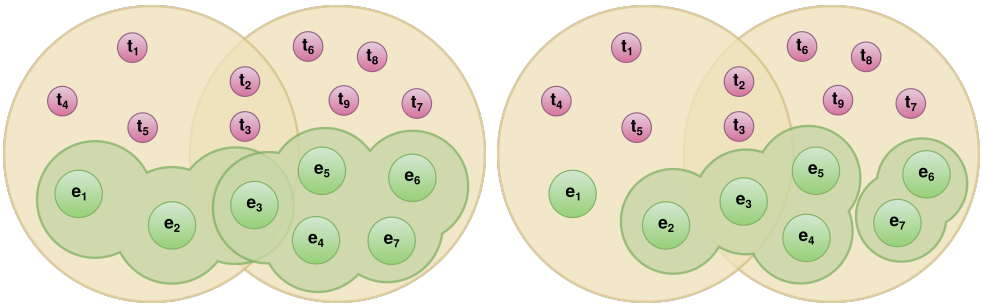
> *Semantic search seeks to improve search [Search Engine Technology] accuracy by understanding the searcher's intent [Intention] and the contextual [Contextual (language use)] meaning of terms as they appear in the searchable dataspace, whether on the Web [World Wide Web] or within a closed system, to generate more relevant results.*
>
> – Semantic search, Wikipedia, 09:10, 7 January 2016

Underlined terms within the text block represent links to other Wikipedia entities (shown in square brackets). This establishes the knowledge block (see Figure 4.1 for a visual illustration). Each term obtained from the tokenization of the text block is represented only once within a document hyperedge, regardless of its frequency within that document. The same happens when multiple links to the same entity are found — the entity is always represented by the same, unique node.

In order to better visualize the differences between this representation and the graph-of-entity, we recommend comparing Figure 7.2 with Figure 6.2, which indexes the same document we illustrate here. For that particular instance of the graph-of-entity, we had 22 *term* nodes and 5 *entity* nodes (the same number as the hypergraph-of-entity), but we also had 32 edges as opposed to only 7 hyperedges in the model we propose here. Such a significant edge reduction was in part possible because of the loss of term dependencies, when switching to the hypergraph-of-entity from the graph-of-entity.

#### 7.4.1.3 *Configurations for related_to hyperedges*

Although *related_to* hyperedges are created based on the triples from the knowledge block in the extended documents, there are several options to group entities according to the provided binary relations (ignoring the predicate). In this work, we explore two approaches for grouping related entities. The first is based on entity co-occurrence, be it within a document or a sentence. This option largely acts as reinforcement, increasing the number of paths leading to the entity, thus improving its chance of being visited. The second is based on a grouping by subject, considering all triples in the collection, similar to the *OutRelations* explored by Neumayer et al. [179]. Accordingly, this approach is more focused on improving connectivity




**(a)** Grouped by co-occurrence. **(b)** Grouped by subject.

**Figure 7.3:** Alternative configurations for the *related_to* hyperedge.

across documents, although it requires more resources during indexing, since it relies on a wide collection view as opposed to a local document view. That is, the generation of *related_to* hyperedges based on grouping by subject will only be complete after all documents in the collection are fully iterated over, while co-occurrence based hyperedges can be computed for each document, independently.

Figure 7.3 illustrates the two approaches. In Figure 7.3a, we find two *related_to* hyperedges, forming the sets $\{e_1, e_2, e_3\}$ and $\{e_3, e_4, e_5, e_6, e_7\}$. As we can see, there is a certain redundancy in defining relations based on entity co-occurrence at the document-level, since the *document* hyperedge already ties the entities. Using sentence-level co-occurrence could be more interesting, however scaling would become an issue, given the much higher number of resulting hyperedges, not to mention the additional preprocessing overhead. It is however clear that the chance to visit nodes like $e_3$ is still reinforced, as is the overall preference for entities as traversal nodes. In analogy to a Wikipedia research task about a given a subject, this would model the higher likelihood of a user following a link to another Wikipedia page (an entity), rather than issuing a new query based on terms extracted from the article.

In Figure 7.3b, we find an alternative approach, where *related_to* hyperedges are used to improve connectivity instead of being used for reinforcement, forming the sets $\{e_2, e_3, e_4, e_5\}$ and $\{e_6, e_7\}$, where one of the entities in the set is the subject in a triple that reaches one of the remaining entities over the whole collection. Traversal-wise, this means that not only can the subject entity reach the object entity, but also that each of the object entities become associated and are now directly reachable through the *related_to* hyperedge. Such a grouping by subject could generate a hyperedge that, in one step, makes a traversal reach a document that would otherwise be multiple degrees apart from the starting node.

While there are multiple approaches for grouping entities based on the provided triples, we only instantiate two in this work, focusing on the grouping by subject, which we use on most of the experiments in this thesis. Specifically, only the hypergraph-of-entity experiments from TREC 2018 Common Core, described in Section 9.2, relies on the document-level co-occurrence *related_to* hyperedges. The experiments described in Sections 8.5, 9.1, 9.3, and B.2, all rely on *related_to* hyperedges grouped by subject.

#### 7.4.1.4 *Extensions*

We provide three types of extensions to the base model — synonyms, context, weights, and TF-bins — which we can combine and reorder in any way we want. Extending the base model with synonymy and contextual relations provides a kind of "organic" query expansion. We usually depend on query expansion to retrieve previously unreachable documents that did not match the user's vocabulary. With the hypergraph-of-entity, this becomes an inherent part of the ranking process, as

it simply requires the addition of new hyperedges linking to related terms. On the other hand, introducing node weights enables term and entity boosting, and introducing hyperedge weights enables document boosting and the assignment of certainty to the information represented by the hyperedge, thus constricting the flow of random walks and directing the walker through the most probable paths. In this section, we provide further details on how synonyms, context, weights and TF-bins were obtained and added to the hypergraph-of-entity.

**SYNONYMS** We used WordNet 3.0, through JWI, the MIT Java Wordnet Interface[1], to obtain the *synset* for the first sense of each term (i.e., the sense that is more frequently used), assuming that the term is a noun. We integrated synonyms into the hypergraph-of-entity by adding missing term nodes (i.e., that were not originally a part of the collection's vocabulary) and linking all terms from the *synset* using a *synonym* hyperedge. For example, if a document contained the term *results*, we would search WordNet as follows:

```
$ wn results -synsn

Synonyms/Hypernyms (Ordered by Estimated Frequency) of
noun result

4 senses of result

Sense 1
consequence, effect, outcome, result, event, issue, upshot
       => phenomenon

Sense 2
solution, answer, result, resolution, solvent
       => statement

Sense 3
result, resultant, final result, outcome, termination
       => ending, conclusion, finish

Sense 4
resultant role, result
       => semantic role, participant role
```

For this particular case, we would obtain four senses. We would then take the synonyms (i.e., the *synset*) from Sense 1 and link all the terms using a *synonym* hyperedge consisting of the following set of terms: *results* (the original term), *consequence*, *effect*, *outcome*, *result*, *event*, *issue* and *upshot*. At the same time, we stored information about the number of senses for each term, as it is useful to compute the weight of the *synonym* hyperedges.

**CONTEXT** Two terms were considered contextually similar whenever they were frequently surrounded by similar terms. We used word2vec [140] word embeddings to establish context, based on the implementation provided by Gensim[2]. The model can either be trained with the same collection that is being indexed, or use an external text collection that might be more relevant to impose context within the given domain. Several hyperparameters can be tuned to control word2vec. We extracted word embeddings of size 100, considering moving windows with 5 words and discarding words with a frequency below 2 in the collection. Once we extracted the embeddings for all terms in the collection, we used the cosine similarity to find the k-nearest neighbors, with $k = 2$, building a similarity network where an edge

1 https://projects.csail.mit.edu/jwi/
2 https://radimrehurek.com/gensim/models/word2vec.html

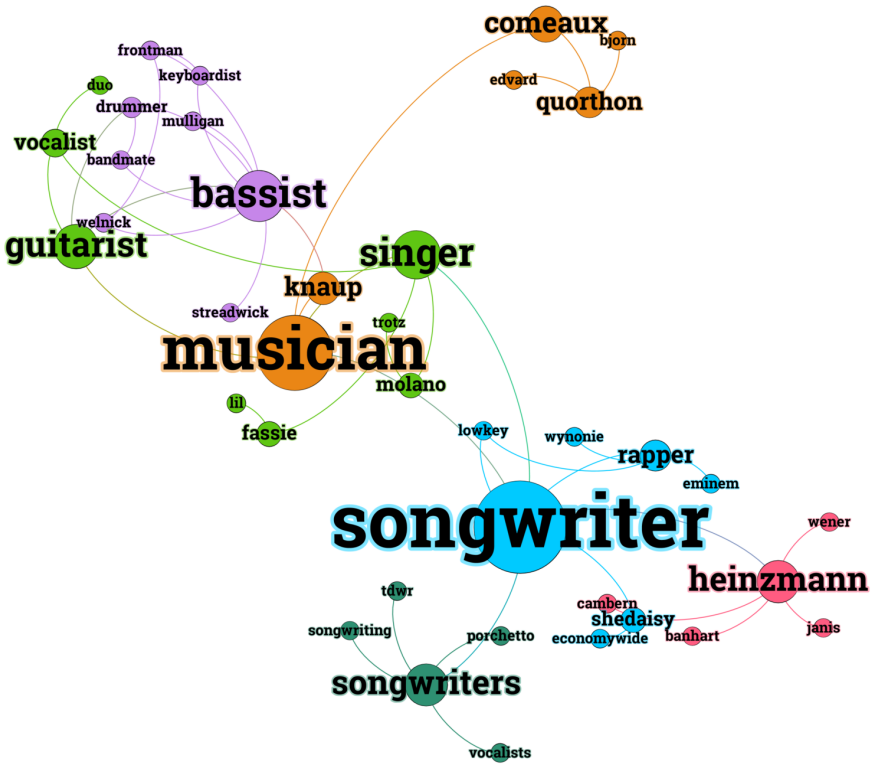



**Figure 7.4:** Word2vec SimNet: *musician* ego network, with a depth of three. Nodes size is proportional to the betweenness centrality and colors identify clusters of densely connected terms.

was created between a term and each of its nearest neighbors, but only when the similarity was above a given threshold (we used 0.5). Throughout this article, we call this the word2vec SimNet. Figure 7.4 shows the neighborhood of *musician* (its context), up to a maximum of three nodes in distance, in the word2vec SimNet for the INEX 2009 Wikipedia subset (see Section 4.1.1). As we can see, even if a query for *guitarist* or *bassist* is issued, documents containing only *musician* can also be considered, although expanding from *guitarist* should result in a higher weight to documents containing *musician* than expanding from *bassist*, since *musician* is adjacent to *guitarist*, but *bassist* can only reach *musician* through *guitarist*. This is the kind of rationale that a graph-based design supports, simultaneously allowing for a better explanation and the promotion of transparency. We integrated this graph-based information into the hypergraph by creating an undirected *context* hyperedge, linking each term to all of its contextually adjacent terms. Were the user to require an explanation as to why a particular ranking was provided for a given query, we would be able to list the paths traversed from the seed nodes representing the query. We could either do it exhaustively (i.e., list all paths), or based on descriptive statistics, like the number of paths leading to ranked nodes, along with a few examples. Either way, graph-based or hypergraph-based models are easily traceable.

Figure 7.5 illustrates the hypergraph-of-entity revision, showing only *synonym* and *context* hyperedges, both examples of n-ary relations between multiple term nodes. We also added any missing term nodes that were external to the document, but present in the list of synonyms or contextually similar terms (in the figure, we only included some of the original terms to illustrate the different patterns). All nodes within blue *context* hyperedges were already a natural part of the hypergraph (i.e., contained in the original collection), since word2vec was trained with the same collection. However, synonyms might be external to the collection, therefore resulting in the addition of new term nodes that are not a part of any document. As we

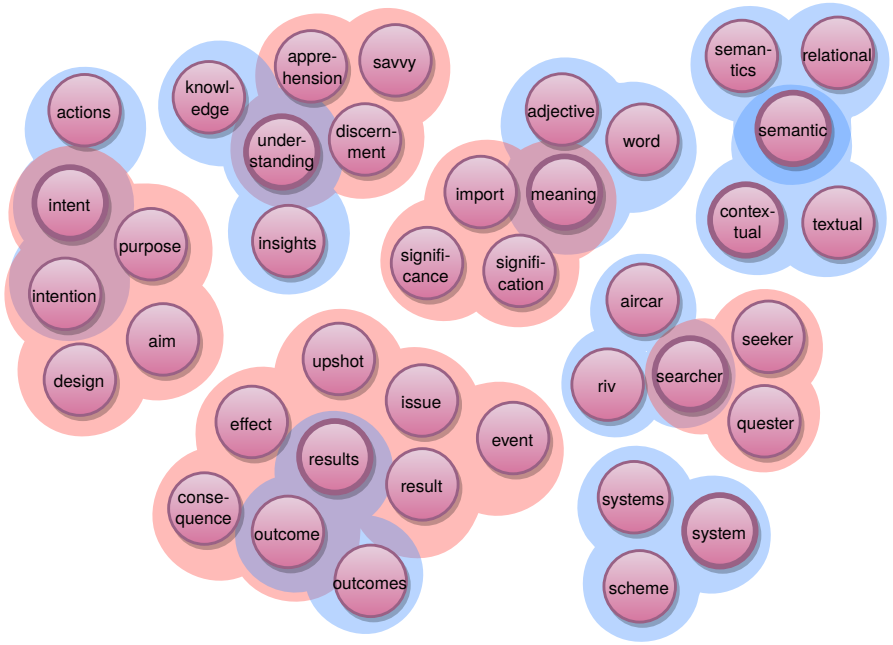



**Figure 7.5:** Hypergraph-of-entity model partial view, showing some of the new *synonym* (red) and *context* (blue) hyperedges. Term nodes from the original document are displayed with a stronger border stroke.

have seen before, both synonyms and contextually similar terms establish bridges between potentially disconnected, but related, documents, increasing the chances of improving recall over the base model. In the figure, nodes that are a part of the document are visually identified by a stronger border. Most of the document term nodes are not synonyms or contextually similar to one another. However, the terms *semantic* and *contextual* are both connected, since *contextual* is one of the top-2 most similar terms to *semantic*, according to their word embeddings. Other interesting subhypergraphs include for instance the neighborhood of term *results*, that contains appropriate synonyms like *consequence*, *result* (the singular) or *effect*, but also less clear synonyms like *issue* or *upshot*; contextually, however, we are able to reach both *outcome* and *outcomes*, with *outcome* already covered by the synonyms (but not its plural), an indicator that relevant related terms might only be reachable through context.

**WEIGHTS** By default, all nodes and hyperedges were unweighted. As another extension to the index, we assigned probabilistic weights to nodes and hyperedges. We did this for two main reasons. First, not all terms or entities (our nodes) are equally relevant, from a query-independent perspective; the same happens for related entities, contextual terms, or synonyms that depend on word sense (our hyperedges). Secondly, assigning weights might serve as a base for pruning in the future, which we predict might improve overall performance. Regarding effectiveness, constricting available paths is a way of increasing focus in the model and thus of guiding random walks. Regarding efficiency, a lower number of nodes and hyperedges result in a lesser amount of used memory, but also in a faster convergence of random walk visit probability, thus requiring less CPU cycles to reach an optimal result. On the other side, the non-uniform random selection of a node or hyperedge during random walks is more expensive than selecting an incident node or hyperedge uniformly at random, which means this requires experimentation.

The aim of the weights assigned to nodes and hyperedges was to provide discriminative power, thus requiring uniform distributions with well dispersed values. In this work, we provide an initial approach to weighting in the hypergraph. Table 7.2 provides an overview of the probabilistic weighting functions that we pro-

**Table 7.2**: Hypergraph-of-entity weighting functions.

**(a)** Nodes.

| Node / Weight | Description |
|---|---|
| *term*<br>$2S\left(\alpha \frac{N - n_t}{n_t}\right) - 1$ | We used a variation of the IDF, with a tunable $\alpha < 1$ parameter to control how fast the function decreases.<br>- S is the sigmoid function<br>- N is the number of documents in the collection<br>- $n_t$ is the number of documents where a given term t occurs.<br>- We used $\alpha = N^{-0.75}$. |
| *entity*<br>Same as *term*. | In the future, we will experiment with different values of $\alpha$ for terms and entities, in particular alternative exponents to $-0.75$. |

**(b)** Hyperedges.

| Hyperedge / Weight | Description |
|---|---|
| *document*<br>0.5 | Linking a term or entity simply through document co-occurrence is weak, so we use a constant weight lower than one. |
| *related_to*<br>$\frac{1}{\lvert e_r \rvert} \sum_{v \in e_r} \frac{\lvert\{u \in e'_r : e'_r \in E_r \setminus \{e_r\} \wedge v \in e'_r\}\rvert}{\lvert e_r \rvert}$ | For each entity within the hyperedge, we calculate the fraction of reachable other entities and average all results.<br>- $E_r$ is the set of all *related_to* hyperedges.<br>- $e_r \in E_r$ is the specific *related_to* hyperedge, for which we are calculating the weight. |
| *contained_in*<br>$\frac{1}{\lvert t \rvert}$ | Links with fewer terms t, where t refers to the tail set in $(t, h) \in E_c \wedge t \subseteq V_t$, should be more frequently followed, since the certainty that the hyperedge leads to the entity is higher. |
| *synonym*<br>$\frac{1}{\lvert e_s \rvert}$ | The higher the number of possible synonyms $e_s \in E_s \wedge e_s \subseteq V_t$, the less certain we are about the hyperedge — we rely on the synonyms of the first (and most probable) sense according to WordNet. |
| *context*<br>$\frac{1}{\lvert e_x \rvert} \sum_{t_i \in e_x \setminus \{t_k\}} \frac{\text{sim}(t_k, t_i) - \text{min}_{\text{sim}}}{1 - \text{min}_{\text{sim}}}$ | A context $e_x \in E_x$ is only as good as the average of all similarities between the original term $t_k \in e_x$ and all other terms $t_i \in e_x \setminus \{t_k\}$. We normalize the weight taking into account the threshold used to create the word2vec SimNet. |

pose, based on the characteristics of each individual node and hyperedge type. For this first experiment with a weighted version of the hypergraph-of-entity, we selected weighting functions that we could compute exclusively using information internal to the model. In an attempt to ensure the generalization of the model, we also restricted the weights to probabilities, in order to facilitate the eventual integration of elements from probabilistic information retrieval or language models in the future.

In particular, for the weighting of terms and entities, we used the probabilistic IDF [321], but replaced the log function with the sigmoid function, to ensure that IDF would always range between zero and one. In the sigmoid IDF we provide a parameter $\alpha$ that controls the function's decrease speed. We manually experimented with multiple values for $\alpha$, finding that the behavior would significantly change for collections of a different dimension. We, therefore, introduced a dependence on a fraction of the collection size N. Figure 7.6 illustrates the behavior of the probabilistic IDF when compared to sigmoid IDF for base N and exponents $-0.5$, $-0.75$ and $-1$. As we can see, using an exponent of $-1$ results in IDF values always being above 0.5 and a slow decrease behavior. On the other hand, using an exponent of $-0.5$ results in a fast decrease with a large fraction of the collection with an IDF closer to zero. Finally, using $-0.75$ results in a decrease speed that is closer to the behavior of the probabilistic IDF assigning a more diverse range of values to different documents in the collection. While we did not specifically tune $\alpha$ to the best approximation to the probabilistic IDF, the value that we selected provides a good enough discriminative power.

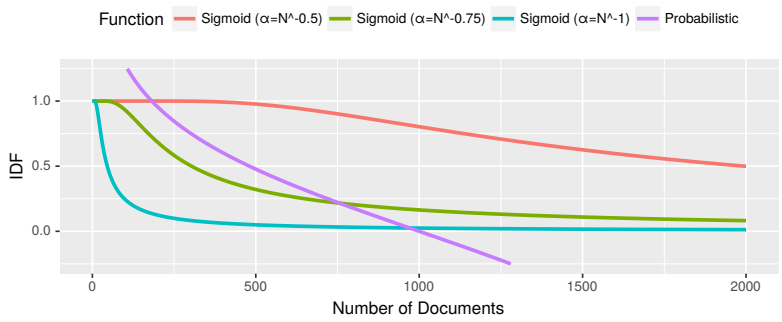



**Figure 7.6:** Selecting $\alpha$ for sigmoid IDF, when compared to the probabilistic IDF.

**Table 7.3:** Term frequency over the complete Wikipedia article on *Semantic search*, from 09:10, 7 January 2016.

**(a)** Terms for 100th percentile on the $]1.5, 54]$ TF interval.

| Term | Frequency |
|---|---|
| search | 54 |
| semantic | 53 |
| web | 9 |
| meaning | 6 |
| system | 4 |
| relevant | 4 |
| results | 4 |
| intent | 3 |
| improve | 2 |
| accuracy | 2 |
| understanding | 2 |

**(b)** Terms for 50th percentile on the $]0, 1.5]$ TF interval.

| Term | Frequency |
|---|---|
| seeks | 1 |
| searcher | 1 |
| contextual | 1 |
| terms | 1 |
| appear | 1 |
| searchable | 1 |
| dataspace | 1 |
| whether | 1 |
| within | 1 |
| closed | 1 |
| generate | 1 |

**TERM FREQUENCY BINS (TF-BINS)** We introduce the concept of TF-bins, which are based on the discretization of the term frequency per document. This way, term frequency can be introduced in the hypergraph-of-entity, while having a low impact in scalability (i.e., we remain focused on forming groups of nodes to minimize the space complexity of the representation model).

For each document, we calculate the term frequency and, for a given number of bins $n$, we compute the percentiles $P_n = \{100\frac{x}{n} \mid x \in \mathbb{Z}^+ \wedge x \leqslant n\}$, assigning them the weight $w(x) = \frac{x}{n}$. So, for example, if we consider $n = 4$ bins, then we compute the percentiles $P_4 = \{25, 50, 75, 100\}$, resulting in four values of TF. Let us for instance consider the following term frequency for 10 documents: $1, 1, 1, 1, 2, 2, 2, 2, 2, 3$. This would result in the value 1 for the 25 percentile, 2 for the 50 and 75 percentiles, and 3 for the 100 percentile. We would then form the TF intervals $]0, 1]$, $]1, 2]$, $]2, 2]$ and $]2, 3]$, with the interval $]2, 2]$ having no matches in $\mathbb{Z}^+$, which prioritizes the lower weights (i.e., we consider $w(x = 2)$, from the 50 percentile, instead of $w(x = 3)$, from the 75 percentile). Per document, and for each non-empty interval, a weighted hyperedge was then created to group terms with a similar term frequency (i.e., within the same TF-bin). This can be used by the ranking function, to issue biased random walks, controlling the flow in a way that the walker will be driven towards documents with a higher TF for the query terms.

Table 7.3 shows the term frequency of all the terms in the example document. While this document was built from the first sentence of Wikipedia's article on *Semantic search*, for illustration purposes we used the term frequencies from the complete document. Our original example would have only contained a single term with $TF = 2$ and the remaining terms would have $TF = 1$, making it less interesting to analyze. Figure 7.7 illustrates the grouping of terms into two TF-bins, as a part of the hypergraph-of-entity. We assigned two different colors to

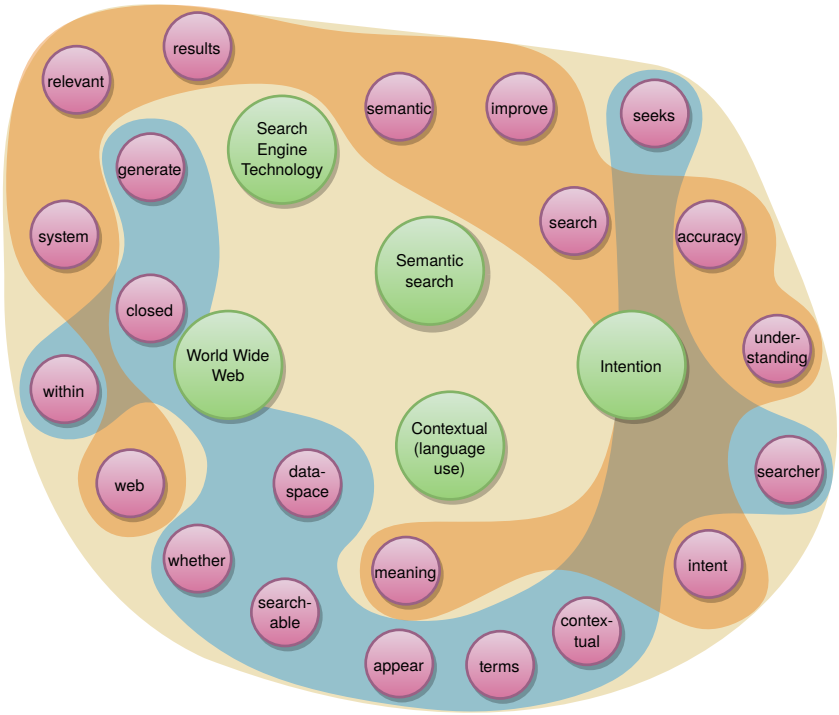



**Figure 7.7:** Hypergraph-of-entity model partial view, showing the *document* hyperedge, along with two *tf_bin* hyperedges (orange for high-TF terms and steel blue for low-TF terms).

*tf_bin* hyperedges, distinguishing between high (orange) and low (steel blue) term frequency bins. The high-TF bins contained terms with term frequencies in the interval $]1.5, 54]$, while the low-TF bins contained terms with term frequencies in the interval $]0, 1.5]$. The *tf_bin* hyperedge for high-TF terms had a weight of $w = 1.0$, while the *tf_bin* hyperedge for low-TF terms had a weight of $w = 0.5$. This bias was introduced to drive the random walkers towards terms that are more probably relevant within the context of each document.

#### 7.4.1.5 *Reflecting on the implications of representation-driven retrieval*

The retrieval models based on indexing data structures like the inverted index are highly dependent on the ranking function that is chosen. These functions usually contain three main elements: (1) term frequency, to measure the importance of query terms in the document; (2) inverse document frequency, to diminish the impact of frequent terms that are widespread over the collection, therefore having little discriminative power; and (3) pivoted document length normalization, to avoid long documents to outrank shorter, more relevant documents. While this is the standard, several other elements can be included based on an inverted index, such as term positions, term boosting payloads, or even document prior features that are query independent and stored in fields of the index. The final score, however, completely depends on the ranking function, that only uses the index to efficiently compute the statistics that it requires.

In the hypergraph-of-entity, however, the graph-based index data structure highly dictates the effectiveness of the ranking function. By reducing the number of possible paths linking terms and entities, as well as ensuring the quality of the retained terms, entities and relations, we increase the chances for the ranking function to succeed. A lower number of possible paths leads to lower uncertainty. In general, the lower the number of documents we consider, the lower the uncertainty as well. This is not to be confused with the quality of the proposed ranking, since smaller collections have a lower probability of having relevant documents for a wide range of

topics. Lower uncertainty is related to the best possible answer the ranking function could return, based on the available information in the index. Given we cannot control the number of documents in a collection, which is often application-dependent, one way to decrease uncertainty is to rely on the reduction of the number of hyperedges and the improvement of node and hyperedge quality. One strategy that we explore to achieve this is the application of keyword extraction. Through it, we also find that lowering the fraction of top keywords improves the effectiveness of the model, just prior to reaching a number that is too low to assure effectiveness (see Table 9.11 for the 0.05 ratio on HGoE, Section 9.3.3).

### 7.4.2 The random walk score as a universal ranking function

We propose a ranking function, based on random walks, that strongly captures the structural features of the hypergraph-of-entity. We compare this function with two baselines from a traditional Lucene[1] inverted index: TF-IDF and BM25 (with default parameters $k_1 = 1.2$ and $b = 0.75$). Both during indexing and querying, text is preprocessed using an analyzer similar to Lucene's *StandardAnalyzer*, with two main differences: (i) stopwords are selected based on the *language-detector* library, using the corresponding dictionaries for the detected language as provided by PostgreSQL 9.6, instead of the default set of English stopwords; (ii) tokens with a length inferior to 3 characters are discarded. The ranking function we propose, random walk score, requires the preselection of a set of seed nodes that represent the query. In this section, we describe the seed node selection process and the random walk score computation approach.

#### 7.4.2.1 *Seed node selection*

The seed node selection process can be seen as part of the "organic" process that enables a kind of stochastic semantic tagging of query parts, akin to named entity recognition in queries. Thus, the first step in calculating the random walk score is to map a keyword query to nodes in the hypergraph-of-entity. This process is similar to the graph-of-entity [268], that is, we tokenize the query into unigrams, mapping them to the corresponding term node (if no match exists, the unigram is simply ignored). The term nodes are then expanded to adjacent entity nodes (the seed nodes), which replace them, unless no adjacent entity node exists, resulting in the term node becoming its own seed node. A confidence weight is then calculated for each seed node, measuring the certainty of the node representing the query. See Devezas et al. [268, Section 3.2, Retrieval] for further details.

Ambiguity is not dealt with during seed node selection, but instead during ranking. During seed node selection, we attempt to reach the whole universe of possibilities (i.e., we find the most complete set of candidate entities that might represent the query). During ranking, however, we rely on the overall relations, naturally stored in the hypergraph, for disambiguation. It is not infrequent to do entity linking based on a graph of entities (and sometimes mentions) and their relations [84, 262, 322]. What we do here is to use basic substring matching to find a large number of candidates (many times we can have over 1,000 candidate nodes per query). Then, during ranking, while capturing the structure of the hypergraph based on random walks, each candidate is visited for a given number of times, depending on the link density of the neighborhood of each seed node. Seed nodes act as an open representation of the query. Ambiguity is then solved by cross-referencing all available information through paths in the graph that depart from the seed nodes. Since we also include synonyms in the hypergraph, we aren't even required to consider multiple word senses, as these are naturally solved based on the knowledge of the model. This is why we simply use substring matching. Although such a naive approach to term-entity linking can be improved, we argue that, based on the

1 https://lucene.apache.org/

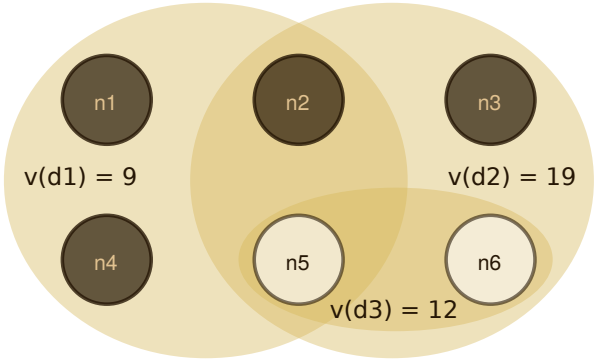



**Figure 7.8:** Hypergraph-of-entity: ranking *document* hyperedges using $RWS(\ell = 2, r = 10)$ for seed nodes $n5$ and $n6$.

described strategy, this is only one step towards entity linking. Moreover, based on the cited literature, this is a step that makes sense for our model, where we already capture links between entities and terms (i.e., mentions), which might even be weighted with different degrees of certainty.

#### 7.4.2.2 *Random walk score*

In random walks, steps can be chosen uniformly at random, but we can also establish a bias through weighted hyperedges (which we can also do for graphs) and weighted nodes (used for a random, non-uniform selection of nodes within a hyperedge). We can also vary the length of the walk $\ell \in \{\ell_1, \ell_2, \ldots, \ell_n\}$, as well as the number of repeats (or iterations) $r \in \{r_1, r_2, \ldots, r_m\}$. In particular, we experimented with the configurations given by $\ell \times r \in \{(\ell_1, r_1), (\ell_1, r_2), \ldots, (\ell_n, r_m)\}$. The goal was to measure the impact of increasing the walk length, the number of repeats, or both. The length $\ell$ constricts or liberates the random walker to wander closer or further apart from the concepts that best represent the query (the seed nodes), while the repeats $r$ improve the certainty of the computed ranking. For evaluation, we used $\ell \in \{2, 3, 4\}$ and $r \in \{10^2, 10^3\}$.

For each seed node, we launched a number $r$ of random walks of length $\ell$, storing the number of visits to each hyperedge. Per seed node, we then normalized the number of visits by dividing by the maximum and then multiplying by the seed node confidence weight. These individual scores were then summed for each hyperedge, obtaining a final document hyperedge score. Random walks respected hyperedge direction, as well as node and hyperedge weights, which introduced bias. In practice, this means that we modeled ad hoc document retrieval as a hyperedge ranking problem using biased random walks for ranking. It also means that a similar strategy can be applied to ad hoc entity retrieval by modeling this task as a node ranking problem instead. This demonstrates how the hypergraph-of-entity is a generalizable model that is easily extensible to other entity-oriented search tasks.

Figure 7.8 illustrates the output of such a random walk, showing three documents represented by their hyperedges, d1, d2 and d3, and six abstract nodes (i.e., they can either represent *term* or *entity* nodes), two of which, $n5$ and $n6$, are identified as seed nodes. We calculated $RWS(\ell = 2, r = 10)$ by launching 10 random walks of length 2 from each of the seed nodes, aggregating the number of visits per hyperedge as $v(d)$. As we can see, both d1 and d2 contain four nodes, overlapping on $n2$ and $n5$, and d2 subsumes d3, that is, it is more general than d3, containing all of its nodes (and more). One of the possible results of the nondeterministic execution of the RWS is show in the figure. Larger values of $r$ (usually $r > 1000$) result in a higher ranking consistency, however, for such a small hypergraph, $r = 10$ is sufficient to converge to the shown ranking: d2, d3, d1.

**Table 7.4:** Mapping entity-oriented search tasks to the hypergraph-of-entity.

| | Query | Input | Results | Output |
|---|---|---|---|---|
| **Ad hoc document retrieval** | Keyword | Term nodes | Documents | Hyperedge ranking |
| **Ad hoc entity retrieval** | Keyword | Term nodes | Entities | Node ranking |
| **Related entity finding** | Entity | One entity node | Entities | Node ranking |
| **Entity list completion** | Entity | Multiple entity nodes | Entities | Node ranking |

**Table 7.5:** Random walk score parameters and chosen configuration.

| Parameter | Description | Configuration |
|---|---|---|
| $\ell$ | Length of the random walk. | 2 |
| $r$ | Number of repeated random walks per seed node. | 10,000 |
| $\Delta_{nf}$ | Number of cycles of node fatigue (see Section B.2). | 0 |
| $\Delta_{ef}$ | Number of cycles of (hyper)edge fatigue (see Section B.2). | 0 |
| *expansion* | Whether to expand query to neighboring entities. | *false* |
| *directed* | Whether to consider or ignore direction. | *true* |
| *weighted* | Whether to consider node and hyperedge weights. | *false* |

#### 7.4.2.3 *Parameterization*

While the hypergraph-of-entity is a representation-driven retrieval model, the ranking function still requires configuration to answer each of the different entity-oriented search tasks. This is achieved both by controlling the type of input and output as described in Table 7.4, and by configuring the parameters from Table 7.5. The random walk score ranking function launches $r$ repeated random walks of a given length $\ell$, from each seed node. Seed nodes can either be the term nodes matching query terms, or their expansion to adjacent entity nodes. Each node and hyperedge has a counter that keeps track of the number of visits from the random walks, which are simulated step by step. This is then used to rank the desired output elements. As we can see in Table 7.4, each of the four main tasks can be mapped to an input/output configuration, thus providing universal ranking for entity-oriented search. For example, by selecting a specific entity as the query (input), and ranking other entities (output), we are able to run the task of related entity finding.

## SUMMARY

In this chapter, we began by arguing the importance of cross-referencing over all available information, for solving general information needs. We have considered the hypergraph data structure as an instrument of generalization, and as an alternative solution for capturing higher-order dependencies in documents, entities and their relations. We then explored the idea of a unified framework for information retrieval, providing clear instances for joint representation approaches and for general retrieval based on a hypergraph. We then proposed a unified model for the representation and retrieval of text and knowledge, supporting multiple tasks of entity-oriented search: ad hoc document retrieval, ad hoc entity retrieval, related entity finding, and entity list completion. The hypergraph-of-entity was proposed as a joint representation of terms, entities and their relations, for indexing corpora and knowledge bases in a unified manner. In this model, entities were linked to other related entities as a group, either according to the knowledge base (e.g., the subject and the respective target objects), or simply the occurrence in a common document; terms and entities were all linked by a document hyperedge, as a bag of words and entities; and terms were linked to entities that they represented or illustrated in some way (e.g., based on string matching with the entity's name; perhaps also good for cross-language retrieval). We presented a base model, as well as multiple combinable extensions, using synonyms provided by WordNet, context provided by word2vec word embeddings similarity, node and hyperedge weighting functions, and the newly introduced concept of TF-bins, based on the discretization of term frequencies. We proposed the random walk score as a retrieval model and as a method for relevance weighting that closely depends on the structure of the hypergraph, which provides the flexibility to change and improve the representation model without the need to repeatedly revise the ranking function. This universal ranking function was based on a series of repeated random walks of a fixed length, starting from a set of seed nodes representing the query. The proposed random walk score was able to cover nodes and hyperedges of all types, but only collected and ranked elements of the selected target type(s). The proposed retrieval model is highly dependent of the representation model, as was well as the random walk score parameters.

# 8 CHARACTERIZING THE HYPERGRAPH-OF-ENTITY REPRESENTATION MODEL

## Contents



Complex networks have frequently been studied as graphs, but only recently has attention been given to the study of complex networks as hypergraphs [323]. As we have seen in Chapter 7, the hypergraph-of-entity is a hypergraph-based model used to represent combined data [92, §2.1.3]. That is, it is a joint representation of corpora and knowledge bases, integrating terms, entities and their relations. It attempts to act as a replacement to solve, by design, the issues of representing combined data through inverted indexes and quad indexes. The hypergraph-of-entity, together with the random walk score, is also an attempt to generalize several tasks of entity-oriented search. This includes ad hoc document retrieval and ad hoc entity retrieval, as well as the recommendation-alike tasks of related entity finding and entity list completion. Since ranking is particularly dependent on the structure of the hypergraph, a characterization is a fundamental step towards improving the representation model and, with it, the retrieval performance.

Accordingly, our focus in this chapter is that of studying the structural features of the hypergraph, following a network science approach, applied to a microscopic and macroscopic scale, as well as over time with an increasing number of documents. This is a task that presents some challenges, both from a practical sense and from a theoretical perspective. While there are many tools [324, 325] and formats [326, 327] for the analysis and transfer of graphs, hypergraphs still lack clear frameworks to perform these functions, making their analysis less trivial. Even formats like GraphML [327] only support undirected hypergraphs. Furthermore, there is still an ongoing study of several aspects of hypergraphs, some of which are trivial in graph theory. For example, the adjacency matrix is a well-established representation of a graph, however recent work is still focusing on defining an adjacency tensor for representing general hypergraphs [115]. As a scientific community,

we have been analyzing graphs since 1735 and, even now, innovative ideas in graph theory are still being researched [250]. However, the concept of hypergraph is much younger, dating from 1970 [40], and thus there are still many open challenges and opportunities. In particular, the main contributions of this chapter are the following:

- Analysis of multiple versions of real-world hypergraph data structures being developed for information retrieval;
- Proposal of a practical analysis framework for hypergraphs;
- Proposal of estimation approaches for the computation of shortest paths and clustering coefficients in hypergraphs;
- Proposal of a computation approach for the density of general mixed hypergraphs based on a corresponding bipartite graph representation;
- Example of an application in the context of information retrieval, where structural features were measured over different hypergraph-based models and presented in relation to the performance of each model.

The structure of this chapter is organized as follows:

- **Section 8.1** begins by providing an overview on the analysis of the inverted index [§8.1.1], knowledge bases [§8.1.2] and hypergraphs [§8.1.3], covering the three main aspects of the hypergraph-of-entity.
- **Section 8.2** describes our characterization approach, covering shortest distance estimation with random walks [§8.2.1] and clustering coefficient estimation with node sampling [§8.2.2], as well as proposing a general mixed hypergraph density formula [§8.2.3] by establishing a parallel with the corresponding bipartite mixed graph.
- **Section 8.3** presents the results of a characterization experiment of the hypergraph–of–entity for the INEX 2009 10T-NL Wikipedia subset.
- **Section 8.4** explores the effect of including synonyms [§8.4.1], contextual similarity [§8.4.2], or TF-bins [§8.4.3] in the structure of the hypergraph.
- **Section 8.5** assesses the retrieval effectiveness of the representation model, analyzing the correlations between the evaluation metrics and the structural features [§8.5.1], and proposing ranking and anomaly indicators based on our conclusions [§8.5.2].

## 8.1 THREE PERSPECTIVES FOR STUDYING THE REPRESENTATION MODEL

The hypergraph-of-entity representation model can be viewed from three different perspectives, which we consider during this analysis. First, it acts as an index and it can be studied accordingly, for example regarding efficiency or space requirements [328, 329]. Secondly, it can be viewed as a knowledge base, where entities and triples can be studied by analyzing the structural features of the RDF graph, for example diameter or density. Thirdly, the representation model can be viewed as a hypergraph, which has special characteristics that a graph does not have, for example hyperedge cardinality. In this section, we explore the statistics and analysis approaches used in the literature, for each of these perspectives.

### 8.1.1 Analyzing inverted indexes

Voorhees [328] compared the efficiency of the inverted index with the top-down cluster search. She analyzed the storage requirements of four test collections, measuring the total number of documents and terms, as well as the average number of terms per document. She then analyzed the disk usage per collection, measuring the number of bytes for document vectors and the inverted index. She measured CPU time in number of instructions and the I/O time in number of data pages accessed at least once, as well as the query time in seconds. Zobel et al. [329] took a similar approach to compare the inverted index and signature files. First, they characterized two test collections, measuring size in mebibytes, number of records and distinct words, as well as the record length, and the number of words, distinct words and distinct words without common terms per record. They also analyzed disk space, memory requirements, ease of index construction, ease of update, scalability and extensibility.

For the hypergraph-of-entity characterization, we focus on studying the structure and size of the hypergraph, as the performance is studied in depth in Chapter 9.

### 8.1.2 Analyzing knowledge bases

Halpin [330] took advantage of Microsoft's *Live.com* query log to reissue queries mentioning entities and concepts over their FALCON-S semantic web search engine. They studied the results, characterizing their sources, triple structure, RDF and OWL classes and properties, and the power-law distributions of the number of URIs, both returned as results and as part of the triples linking to the results. They focused mostly on measuring the frequency of different elements or aggregations (e.g., top-10 domain names for the URIs, most common data types, top vocabulary URIs).

Ge et al. [331] studied the projection of tripartite RDF graphs, defining an object link graph based on paths linking objects (i.e., target node entities), either directly or through blank nodes. They studied the Falcons Crawl 2008 and 2009 datasets (FC08 and FC09), which included URLs from domains like bio2rdf.org or dbpedia.org. They characterized each object link graph based on density (extrapolated from the average degree), connectivity, largest connected component, and diameter. By comparing the object link graphs for the FC08 and FC09, they were able to characterize the structural evolution of the object link graph. They also analyzed two domain-specific subgraphs (according to URL domains) from FC09, comparing each subgraph with its original graph. Comparing two snapshots of the same data enabled them to find evidence of the scale-free nature of the network. While the graph almost doubled in size from FC08 to FC09, the average degree remained the same and the diameter actually decreased.

Fernandez et al. [332] focused on studying the structural features of RDF data. They proposed several subject and object degrees, accounting for the number of links from/to a given subject/object (outdegree and indegree), the number of links from a $\langle subject, predicate \rangle$ (partial outdegree) and to a $\langle predicate, object \rangle$ (partial indegree), the number of distinct predicates from a subject (labeled outdegree) and to an object (labeled indegree), and the number of objects linked from a subject through a single predicate (direct outdegree), as well as the number of subjects linking to an object through a single predicate (direct indegree). They also measured predicate degree, outdegree and indegree. They proposed common ratios to account for shared structural roles of subjects, predicates and objects (e.g., subject-object ratio). Global metrics were also defined for measuring the maximum and average outdegree of subject and object nodes for the whole graph. Another analysis approach was focused on the predicate lists per subject, measuring the ratio of repeated lists and their degree, as well as the number of lists per predicate. Finally, they also de-

fined several statistics to measure typed subjects and classes, based on the *rdf:type* predicate.

While we study a hypergraph that jointly represents terms, entities and their relations, we focus on a similar characterization approach, that is more based on structure and less based on measuring performance.

### 8.1.3 Analyzing hypergraphs

Hypergraphs [40] have been around since 1970. While this concept was introduced by Claude Berge on this year, there had been other contributions surrounding the topic, namely in extremal graph and set theory. Post-1970, the work by Erdös [333] and Brown et al. [334] illustrates the intersection between extremal graph theory and hypergraph theory, while, pre-1970, we can also find contributions like Sperner's theorem [335], in extremal set theory, or the Turán number [336, 337], in extremal graph theory. Interestingly, hypergraphs have remained somewhat fringe in network science, perhaps due to Paul Erdös resistance to the concept [40]:

> *At the Balatonfüred Conference (1969), P. Erdös and A. Hajnal asked us why we would use hypergraphs for problems that can be also formulated in terms of graphs. The answer is that by using hypergraphs, one deals with generalizations of familiar concepts. Thus, hypergraphs can be used to simplify as well as to generalize.*

Although Erdös himself, who was interested in exploring the representation of graphs using set intersections [338], also studied hypergraph problems, he avoided this designation, only sparsely using it [334]:

> *By an r-graph we mean a fixed set of vertices together with a class of unordered subsets of this fixed set, each subset containing exactly r elements and called an r-tuple. In the language of Berge [40] this is a simple uniform hypergraph of rank r.*

Hypergraphs are data structures that can capture higher-order relations. As such, they either present conceptually different or multiple counterparts to the equivalent graph statistics. Take for instance the node degree. While graphs only have a node degree, indegree and outdegree, hypergraphs can also have a hyperedge degree, which is the number of nodes in a hyperedge [339]. The hyperedge degree also exists for directed hyperedges, in the form of a tail degree and a head degree[1]. The tail degree is based on the cardinality of the source node set and the head degree is based on the cardinality of the target node set. In this work we rely on the degree, clustering coefficient, average path length, diameter and density to characterize the hypergraph-of-entity.

Building on the work by Gallo et al. [340], who extended Dijkstra's algorithm to hypergraphs, and the work by Ausiello et al. [341], who tackled the same problem using a dynamic approach, Gao et al. [342] have also proposed two algorithms for computing shortest paths in hypergraphs. The first, HyperEdge-based Dynamic Shortest Path (HE-DSP), like Gallo et al., proposed an extension to Dijkstra's algorithm. The second, Dimension Reduction Dynamic Shortest Path (DR-DSP), relied on an induced graph with the same vertex set, adding weighted edges when a hyperedge containing the two vertices exists in the corresponding hypergraph, while selecting the minimum weight over all available hyperedges for the pair of vertices.

In this work, we focus on approximated computation approaches, which are useful for large-scale hypergraphs. Ribeiro et al. [343] proposed the use of multiple random walks to find shortest paths in power law networks. They found that random walks had the ability to observe a large fraction of the network and that two

1 Tail and head is used in analogy to an arrow, not a list.

random walks, starting from different nodes, would intersect with a high probability. Glabowski et al. [344] contributed with a shortest path computation solution based on ant colony optimization, clearly structuring it as pseudocode, while providing several configuration options. Parameters included the number of ants, the influence of pheromones and other data in determining the next step, the speed of evaporation of the pheromones, the initial, minimum and maximum pheromone levels, the initial vertex and an optional end vertex. Li [345] studied the computation of shortest paths in electric networks based on random walk models and ant colony optimization, proposing a current reinforced random walk model inspired by the previous two. In this work, we also use a random walk based approach to approximate shortest paths and estimate the average path length and diameter of the graph.

Gallagher and Goldberg [316, Eq.4] provide a comprehensive review on clustering coefficients for hypergraphs. The proposed approach for computing the clustering coefficient in hypergraphs accounted for a pair of nodes, instead of a single node, which is more frequent in graphs. Based on these two-node clustering coefficients, the node cluster coefficient was then calculated. Two-node clustering coefficients measured the fraction of common hyperedges between two nodes, through the intersection of the incident hyperedge sets for the two nodes. It then provided different kinds of normalization approaches, either based on the union, the maximum or minimum cardinality, or the square root of the product of the cardinalities of the hyperedge sets. The clustering coefficient for a node was then computed based on the average two-node clustering coefficient for the node and its neighbors.

The codegree Turán density [346] $\gamma(\mathcal{F})$ can be computed for a family $\mathcal{F}$ of $k$-uniform hypergraphs, also known as $k$-graphs. It is calculated based on the codegree Turán number $\text{co-ex}(n, \mathcal{F})$ — the extremal number based on the codegree of a hypergraph — which takes as parameters the number of nodes $n$ and the family $\mathcal{F}$ of $k$-graphs. In turn, the codegree Turán number is calculated based on the minimum number of nodes, taken from all sets of $r-1$ vertices of each hypergraph $H_n$ that, when united with an additional vertex, form a hyperedge from $H$. The codegree density for a family $\mathcal{F}$ of hypergraphs is then computed based on $\limsup_{n\to\infty} \frac{\text{co-ex}(n,\mathcal{F})}{n}$. Since this was the only concept of density we found associated with hypergraphs or, more specifically, a family of $k$-uniform hypergraphs, we opted to propose our own density formulation (Section 8.2). Furthermore, the hypergraph-of-entity is a single general mixed hypergraph. In other words, it is not a family of hypergraphs, it contains hyperedges of multiple degrees (it's not $k$-uniform, but general) and it contains undirected and directed hyperedges (it's mixed). Accordingly, we propose a density calculation based on the counterpart bipartite graph of the hypergraph, where hyperedges are translated to bridge nodes.

## 8.2 HYPERGRAPH CHARACTERIZATION APPROACH

Graphs can be characterized at a microscopic, mesoscopic and macroscopic scale. The microscopic analysis is concerned with statistics at the node-level, such as the degree or clustering coefficient. The mesoscopic analysis is concerned with statistics and patterns at the subgraph-level, such as communities, network motifs or graphlets. The macroscopic analysis is concerned with statistics at the graph-level, such as average clustering coefficient or diameter. In this work, our analysis of the hypergraph is focused on the microscopic and macroscopic scales. We compute several statistics for the whole hypergraph, as well as for snapshot hypergraphs that depict growth over time. Some of these statistics are new to hypergraphs, when compared to traditional graphs. For instance, nodes in directed graphs have an indegree and an outdegree. However, nodes in directed hypergraphs have four degrees, based on incoming and outgoing nodes, as well as on incoming and outgoing

hyperedges. While in graphs all edges are binary, leading to only one other node, in hypergraphs hyperedges are n-ary, leading to multiple nodes, and thus different degree statistics. While some authors use 'degree' to refer to node and hyperedge degrees [347, §4][339, §Network Statistics in Hypergraphs], in this work we opted to use the 'degree' designation when referring to nodes and the 'cardinality' designation when referring to hyperedges. This is to avoid any confusion for instance between an "hyperedge-induced" node degree and a hyperedge cardinality.

We analyze the base model, as well as three models based on the synonyms, contextual similarity and TF-bins extensions. For the full hypergraph of each of the four models, we compute the following global statistics:

- Number of nodes, in total and per type;
- Number of hyperedges, in total, per direction, and per type;
- Average degree;
- Average clustering coefficient;
- Average path length;
- Diameter;
- Density.

We also plot the following distributions for the full hypergraph:

- Node degree distributions per node type:
  - Node-based node degree;
  - Hyperedge-based node degree.
- Hyperedge cardinality distributions per hyperedge type.

Then, we define a temporal analysis framework based on an increasing number of documents (i.e., time passes as documents are added to the hypergraph-of-entity index). We prepare several snapshots, with a different number of documents each, for each of the four models. We then compute and plot the following statistics for each snapshot, showing its evolution as the number of documents increases:

- Average node degree over time;
- Average hyperedge cardinality over time;
- Average diameter and average path length over time;
- Average clustering coefficient over time;
- Average density over time.
- Size over time:
  - Number of nodes;
  - Number of hyperedges;
  - Space in disk;
  - Space in memory.

Finally, we also measure the run time for several operations, in order to understand the efficiency cost and the evolution of its behavior for an increasing number of documents:

- Index creation time;
- Global statistics computation time;
- Node degrees computation time;
- Hyperedge cardinalities computation time.

In order to support large-scale hypergraphs, we compute the average path length, diameter, clustering coefficient, and density using approximated strategies. We estimate shortest distances based on random walks, the clustering coefficient based on node sampling, and the density based on a bipartite graph induced from the hypergraph, although without the need to explicitly create this graph. The following sections detail these approaches.

### 8.2.1 Estimating shortest distances with random walks

Ribeiro et al. [343] found that, in power law networks, there is a high probability that two random walk paths, usually starting from different nodes, will intersect and share a small fraction of nodes. We took advantage of this conclusion, adapting it to a hypergraph, in order to compute a sample of shortest paths and their length, used to estimate the average path length and diameter. We considered two (ordered) sets $S_1 \subset V$ and $S_2 \subset V$ of nodes sampled uniformly at random, each of size $s = |S_1| = |S_2|$. We then launched $r$ random walks of length $\ell$ from each pair of nodes $S_1^i$ and $S_2^i$. For a given pair of random walk paths, we iterated over the nodes in the path starting from $S_1^i$, until we found a node in common with the path starting from $S_2^i$. At that point, we merged the two paths based on the common node, discarding the suffix of the first path and the prefix of the second path. We computed the length of these paths, keeping only the minimum length over the $r$ repeats. As the number of iterations $r$ increased, we progressively approximated the shortest path for the pair of nodes. Despite the inherent estimation error, this method can be used to study even large-scale hypergraphs — precision can be controlled by tuning the number of sampled nodes and random walks, eventually leading to convergence for large values. This approach enabled us to generate a sample of approximated shortest path lengths, which could be used to compute the estimated diameter (its maximum) and the estimated average path length (its mean), in a scenario where high precision is not critical. This is true for instance for a quick or initial analysis of a hypergraph. Given the repeated research iterations over the hypergraph-of-entity and the multitude of tests carried over this model, a quick estimation approach is ideal.

### 8.2.2 Estimating clustering coefficients with node sampling

In a graph, the clustering coefficient is usually computed for a single node and averaged over the whole graph. As shown by Gallagher and Goldberg [316, §I.A.], in hypergraphs the clustering coefficient is computed, at the most atomic level, for a pair of nodes. The clustering coefficient for a node is then computed based on the averaged two-node clustering coefficients between the node and each of its neighbors (cf. Gallagher and Goldberg [316, Eq.4]). Three options were provided for calculating the two-node clustering coefficient, one of them based on the Jaccard index between the neighboring hyperedges of each node [316, Eq.1], which we use in this work. While a global understanding of the clustering coefficient is useful for characterizing the overall local connectivity in the hypergraph, the existence of a random hypergraph generation model, like the Watts–Strogatz model [348] for graphs, would provide further interpretations at a mesoscale. We leave this open and instead focus on the macroscale.

Continuing with the philosophy of large-scale hypergraph support in our analysis framework, as opposed to computing the clustering coefficient for all nodes, we estimated the clustering coefficients for a smaller sample $S \subseteq V$ of nodes. Furthermore, for each sampled node $s_i \in S$, we also sampled its neighbors $N_S(s_i)$ for computing the two-node clustering coefficients. We then applied the described equations to obtain the clustering coefficients for each node $s_i$ and a global clustering coefficient based on the overall average. For $S = V \wedge N_S(s_i) = N(s_i)$, being $N_S$ the sampled neighbors and $N$ the full set of neighbors, we would obtain the exact clustering coefficient. Again, this approach offers two parameters that can be controlled as a tradeoff between efficiency and effectiveness.

### 8.2.3 Computing the density of general mixed hypergraphs

A general mixed hypergraph is general (or non-uniform) in the sense that its hyperedges can contain an arbitrary number of vertices, and it is mixed in the sense that it can contain hyperedges that are either undirected and directed. We compute a hypergraph's density by analogy with its corresponding bipartite graph, which contains all nodes from the hypergraph, along with connector nodes representing the hyperedges.

Consider the hypergraph $H = (V, E)$, with $n = |V|$ nodes and $m = |E|$ hyperedges. Also consider the set of all undirected hyperedges $E_U$ and directed hyperedges $E_D$, where $E = E_U \cup E_D$. Their subsets $E_U^k$ and $E_D^{k_1,k_2}$ should also be respectively considered, where $E_U^k$ is the subset of undirected hyperedges with $k$ nodes and $E_D^{k_1,k_2}$ is the subset of directed hyperedges with $k_1$ tail (source) nodes, $k_2$ head (target) nodes and $k = k_1 + k_2$ nodes, assuming the hypergraph only contains directed hyperedges between disjoint tail and head sets. This means that the union of $E_U = E_U^1 \cup E_U^2 \cup E_U^3 \cup \ldots$ and $E_D = E_D^{1,1} \cup E_D^{1,2} \cup E_D^{2,1} \cup E_D^{2,2} \cup \ldots$ forms the set of all hyperedges $E$. We use it as a way to distinguish between hyperedges with different degrees. This is important because, depending on the degree $k$, the hyperedge contributes differently to the density, when considering the corresponding bipartite graph. For instance, one undirected hyperedge with degree $k = 4$ contributes with four edges to the density. Accordingly, we derive the density of a general mixed hypergraph as shown in Equation 8.1.

$$D = \frac{2\sum_k k|E_U^k| + \sum_{k_1,k_2}(k_1 + k_2)|E_D^{k_1,k_2}|}{2(n+m)(n+m-1)} \tag{8.1}$$

In practice, this is nothing more than a comprehensive combination of the density formulas for undirected and directed graphs. On one side, we consider the density of a mixed graph that should result of the combination of an undirected simple graph and a directed simple graph. That is, each pair of nodes can be connected, at most, by an undirected edge and two directed edges of opposing directions. On the other side, we use hypergraph notation to directly obtain the required statistics from the corresponding mixed bipartite graph, thus calculating the analogous density for a hypergraph.

## 8.3 ANALYZING THE HYPERGRAPH-OF-ENTITY

We indexed a subset of the INEX 2009 Wikipedia collection [112] given by the 7,487 documents appearing in the relevance judgments of 10 random topics. We then computed global statistics (macroscale), local statistics (microscale) and temporal statistics. Temporal statistics were based on an increasingly larger number of documents, by creating several snapshots of the index, through a 'limit' parameter, until all documents were considered.

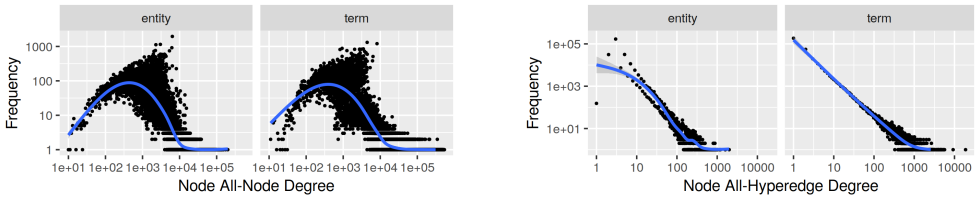
**Table 8.1:** Global statistics for the base model.

| Statistic | Value |
|---|---|
| Nodes | 607,213 |
| *term* | 323,672 |
| *entity* | 283,541 |

| Statistic | Value |
|---|---|
| Hyperedges | 253,154 |
| Undirected | 14,938 |
| *document* | 7,484 |
| *related_to* | 7,454 |
| Directed | 238,216 |
| *contained_in* | 238,216 |

| Statistic | Value |
|---|---|
| Avg. Degree | 0.8338 |
| Avg. Clustering Coefficient | 0.1148 |
| Avg. Path Length | 8.3667 |
| Diameter | 17 |
| Density | 3.88e-06 |

**Figure 8.1:** Node degree distributions for the base model (log-log scale).




**(a)** Based on connected nodes.

**(b)** Based on connected hyperedges.

**GLOBAL STATISTICS** In Table 8.1, we present several global statistics about the hypergraph-of-entity, in particular the number of nodes and hyperedges, discriminated by type, the average degree, the average clustering coefficient, the average path length, the diameter and the density. The average clustering coefficient was computed based on a sample of 5,000 nodes and a sample of 100,000 neighbors for each of those nodes. The average path length and the diameter were computed based on a sample of shortest distances between 30 random pairs of nodes and the intersections of 1,000 random walks of length 1,000 launched from each element of the pair. Finally, the density was computed based on Equation 8.1. As we can see, for the 7,487 documents the hypergraph contains 607,213 nodes and 253,154 hyperedges of different types, an average degree lower than one (0.83) and a low clustering coefficient (0.11). It is also extremely sparse, with a density of 3.9e−06. Its diameter is 17 and its average path length is 8.4, almost double when compared to a social network like Facebook [349].

**LOCAL STATISTICS** Figure 8.1 illustrates the node degree distributions. In Figure 8.2a, the node degree is based on the number of connected nodes, with the distribution approximating a log-normal behavior. In Figure 8.2b, the node degree is based on the number of connected hyperedges, with the distribution approximating a power law. This shows the usefulness of considering both of the node degrees in the hypergraph-of-entity, as they are able to provide different information.

Figure 8.2 illustrates the hyperedge cardinality distribution. For *document* hyperedges, cardinality is log-normally distributed, while for *related_to* hyperedges the behavior is slightly different, with low cardinalities having a higher frequency than they would in a log-normal distribution. Finally, the cardinality distribution of *contained_in* hyperedges, while still heavy-tailed, presents an initial linear behavior, followed by a power law behavior. The maximum cardinality for this type of hyperedge is also 16, which is a lot lower when compared to *document* hyperedges and *related_to* hyperedges, which have cardinality 8,167 and 3,084, respectively. This is explained by the fact that *contained_in* hyperedges establish a directed connection between a set of terms and an entity that contains those terms, being limited by the maximum number of words in an entity.

**TEMPORAL STATISTICS** In order to compute temporal statistics, we first generated 14 snapshots of the index based on a limit L of documents, for $L \in \{1, 2, 3, 4, 5, 10, 25, 50, 100, 1000, 2000, 3000, 5000, 8000\}$.

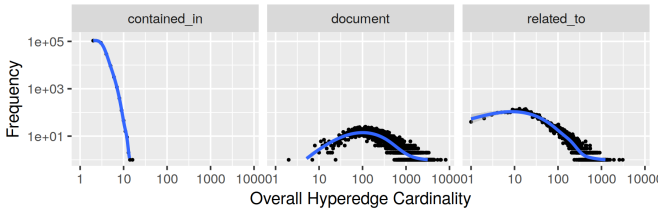
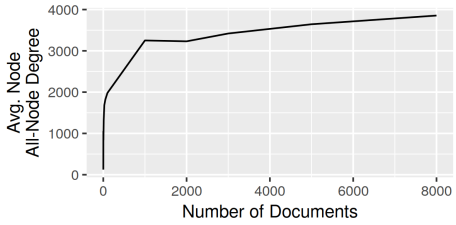
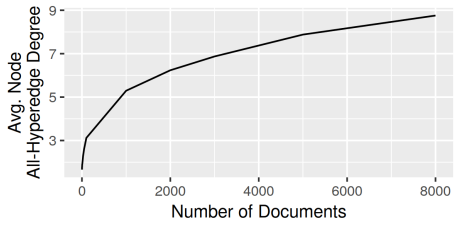



**Figure 8.2:** Hyperedge cardinality distribution based on the total number of nodes for the base model (log-log scale).

**Figure 8.3:** Average node degree over time for the base model.




**(a)** Based on connected nodes.




**(b)** Based on connected hyperedges.

Figure 8.3 illustrates the node-based and hyperedge-based average node degrees over time (represented as the number of documents in the index at a given instant). As we can see, both functions tend to converge, however this is clearer for the node-based degree, reaching nearly 4,000 nodes, through only 9 hyperedges, on average. Figure 8.4 illustrates the average undirected hyperedge cardinality over time, with a convergence behavior that approximates 300 nodes per hyperedge, after rising to an average of 411.88 nodes for $L = 25$ documents.

Figure 8.5 illustrates the evolution of the average path length and the diameter of the hypergraph over time. For a single document, these values reached 126.1 and 491, respectively, while, for just two documents, they immediately lowered to 3.8 and 10. For higher values of L, both statistics increased slightly, reaching 7.2 and 15 for the maximum number of documents. Notice that these last values are equivalent to those computed in Table 8.1 (8.4 and 17, respectively), despite resulting in different amounts. This is due to the precision errors in our estimation approach, resulting in a difference of 1.2 and 2, respectively, which is tolerable when computation resources are limited. In Figure 8.6, we illustrate the evolution of the clustering coefficient, which rapidly decreases from 0.59 to 0.11. The low average path length and clustering coefficient point towards a weak community structure, possibly due to the coverage of diverse topics. However, we would require a random hypergraph generation model, like the Watts–Strogatz model [348] for graphs, in order to properly interpret the statistics.

Figure 8.7 illustrates the evolution of the density over time. The density is consistently low, starting from 1.37e−03 and progressively decreasing to 3.91e−06 as the number of documents increases. This shows that the hypergraph-of-entity is an extremely sparse representation, with limited connectivity, which might benefit precision in a retrieval task.

Figure 8.8 displays the number of nodes (8.9a) and hyperedges (8.9b) created over time, as the index grew. Both presented a sub-linear growth behavior, reaching 4,566 nodes and 803 hyperedges for 10 documents, 238,141 nodes and 89,348 hyperedges for 2,000 documents, and 607,213 nodes and 253,154 for the whole collection of 7,487 documents. The ratio of hyperedges per node evolved from 0.18, to 0.38, to 0.42, always staying below one. This means that the number of hyperedges increased slower than the number of nodes. Moreover, we know that nodes represent terms and entities, which will eventually converge to a finite vocabulary, further decreasing index growth rate.

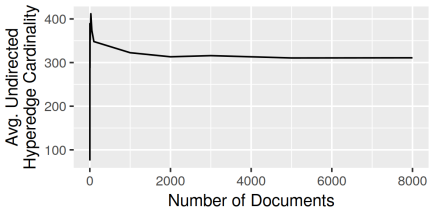



**Figure 8.4:** Average hyperedge cardinality over time for the base model.

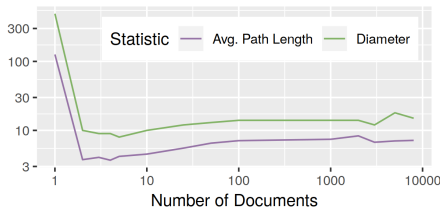



**Figure 8.5:** Average estimated diameter and average shortest path over time for the base model.

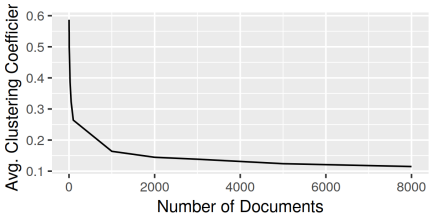



**Figure 8.6:** Average estimated clustering coefficient over time for the base model.

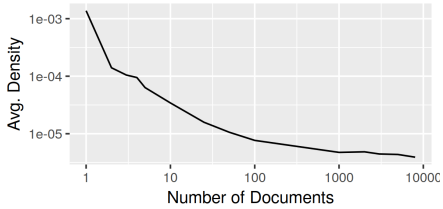


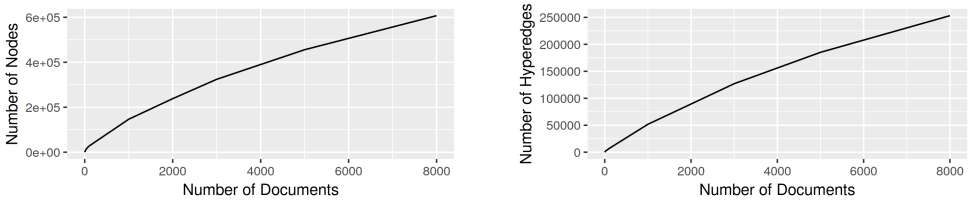

**Figure 8.7:** Average density over time for the base model.

As shown in Figure 8.9, we also measured the space usage of the hypergraph, both in disk (8.10a) and in memory (8.10b). In disk, the smallest snapshot required 43.8 KiB for one document, while the largest snapshot required 181.9 MiB for the whole subset. Average disk space over all snapshots was 37.5 MiB $\pm$ 58.9 MiB. In memory, for our particular application[1], the smallest snapshot used 1.0 GiB for one document, including the overhead of the data structures, and the largest snapshot used 2.3 GiB for the whole subset. Average memory space over all snapshots was 1.3 GiB $\pm$ 461.1 MiB. Memory also grew faster for the first 1,000 documents, apparently leading to an expected convergence, although we could not observe it for such a small subset.

Finally, Figure 8.10 illustrates the base model run times of the following operations for an increasing number of documents: index creation (8.10a); the computation of the global statistics (8.10b), also shown in Table 8.1; the computation of all node degrees (8.10c); and the computation of all hyperedge cardinalities (8.10d). As we can see, the most significant increase in run time happens around 1,000 documents, with the exception of the global statistics computation, which shows an increased run time for the first added documents. A possible reason for this anomaly is that this is the first analysis operation that we run after creating the index, which might influence the caching mechanisms of the system, thus reducing run time after the first documents and then resuming regular behavior. Indexing time took 1m09s for 1,000 documents and 4m13s for a maximum of 8,000 documents. The computation of global statistics took 17m26s for 1,000 documents and 41m18s for a maximum of 8,000 documents. Node degrees were computed in 4m27s for 1,000 documents, taking 20m55s at most, while hyperedge cardinalities were computed in only 19s for 1,000 documents, taking 44s at most, making it the most efficient statistic to compute.

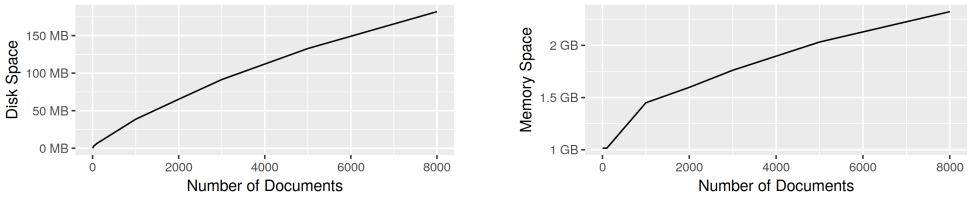
1 We relied on the Grph Java library, available at http://www.i3s.unice.fr/~hogie/software/index.php?name=grph, to represent the hypergraph in memory.

**Figure 8.8:** Number of nodes and hyperedges over time for the base model.




**(a)** Nodes. **(b)** Hyperedges.

**Figure 8.9:** Required space for storing and loading the base model over time.




**(a)** Disk. **(b)** Memory.

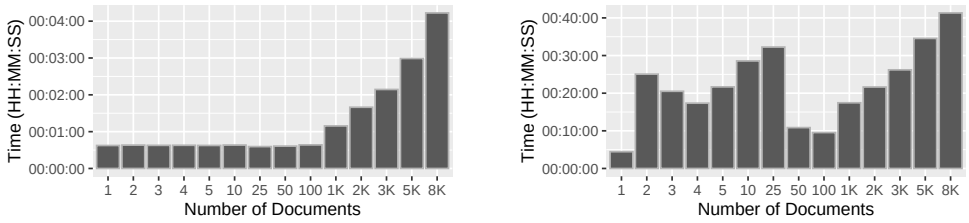
## 8.4 ANALYZING THE STRUCTURAL IMPACT OF DIFFERENT INDEX EXTENSIONS

In this section, we continue the characterization work by taking into consideration the index extensions, applied over the hypergraph-of-entity base model, as described in Section 7.4.1.4. In Sections 8.4.1 and 8.4.2, we study the structural impact of synonyms and context, respectively. In Section 8.4.3, we study the structural impact of the TF-bins index extension, while also considering different numbers of bins.

### 8.4.1 Synonyms

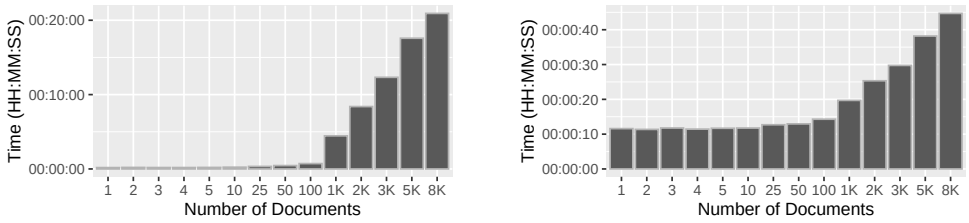
The base model for the hypergraph-of-entity establishes n-ary connections, both directed and undirected, among nodes that represent terms and entities. Most visibly, *document* hyperedges group all terms and entities mentioned in a document, a lot like a bag of words and entities that integrates both unstructured and structured evidence. This model can easily be extended with synonyms, that establish new bridges between documents. In particular, we used the synsets from WordNet 3.0 [304], based on the first sense of each term in the hypergraph, and only considering its noun form. Each synset was modeled as a *synonym* hyperedge. In this section, we characterize the hypergraph-of-entity when using the synonyms extension. We repeat the analysis described in Section 8.3, but only cover results that show a different behavior from the base model.

Table 8.2 shows the global statistics for the synonyms model. As we can see, the number of terms increased from 323,672 (cf. Table 8.1) to 326,671. This means that 2,999 synonym terms that did not originally belong to the collection were added. The number of undirected hyperedges increased significantly, with 10,650 new synonymy relations. The average degree slightly increased, with the average clustering coefficient and the density remaining stable. The diameter also remained at 17, however the average path length decreased almost a unit, from 8.37 to 7.53, approximating nodes through the relation of synonymy. This is an indicator of the usefulness of using synonyms to establish new bridges between documents. In fact, we found 4,558 new paths created by this extension, resulting in 65.29 documents




**(a)** Index creation.

**(b)** Global statistics computation.




**(c)** Node degrees computation.

**(d)** Hyperedge cardinalities computation.

**Figure 8.10:** Base model run time statistics.

**Table 8.2:** Global statistics for the synonyms model.

| Statistic | Value |
|---|---|
| Nodes | 610,212 |
| *term* | 326,671 |
| *entity* | 283,541 |

| Statistic | Value |
|---|---|
| Hyperedges | 263,804 |
| Undirected | 25,588 |
| *document* | 7,484 |
| *related_to* | 7,454 |
| *synonym* | 10,650 |
| Directed | 238,216 |
| *contained_in* | 238,216 |

| Statistic | Value |
|---|---|
| Avg. Degree | 0.8646 |
| Avg. Clustering Coefficient | 0.1168 |
| Avg. Path Length | 7.5333 |
| Diameter | 17 |
| Density | 3.88e-06 |

linked on average per synonym. Besides global statistics, we also identified four interesting changes or new characteristics when compared to the base model:

- Term node degree distribution;
- Synonym hyperedge cardinality distribution;
- Average hyperedge cardinality over time;
- Average estimated diameter and average path length over time.

**TERM NODE DEGREE DISTRIBUTION** Figure 8.11 illustrates the node-based node degree distribution for entity and term nodes in the hypergraph-of-entity with the synonyms extension. While the behavior for entity nodes is similar to the base model, term nodes show a combination of a power law like behavior for the lower

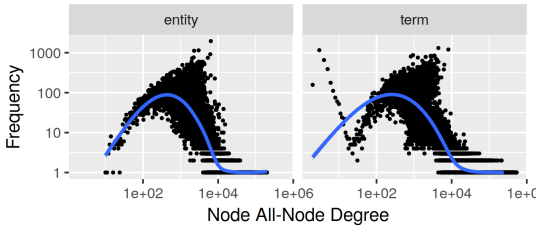



**Figure 8.11:** Node-based node degree distribution, for the synonyms model (log-log scale).

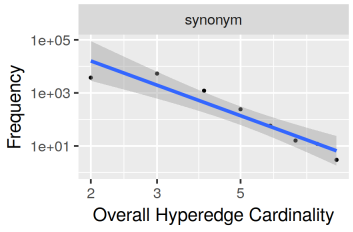



**Figure 8.12:** Synonym hyperedge cardinality distribution (log-log scale).

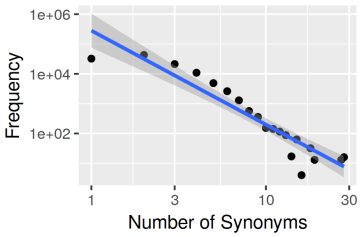



**Figure 8.13:** WordNet 3.0 noun synonyms distribution (log-log scale).

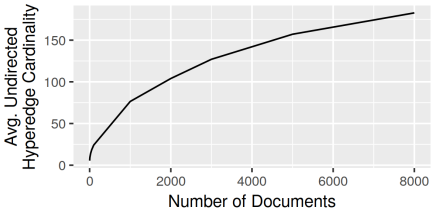



**Figure 8.14:** Average hyperedge cardinality over time for the synonyms model.

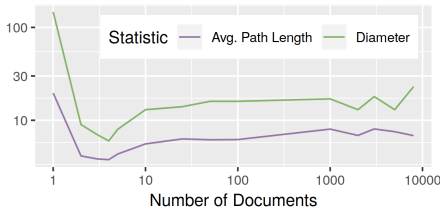



**Figure 8.15:** Average estimated diameter and average shortest path over time for the synonyms model.

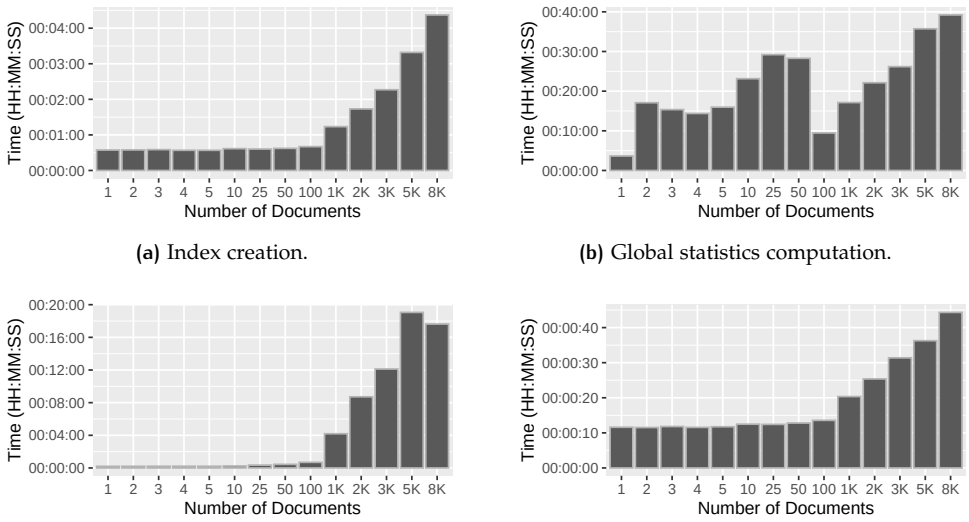
degrees, with a log-linear behavior for the remaining degrees. This is due to the introduction of synonyms from WordNet, which, as we can see in Figure 8.13, follow a distribution close to a power law.

**SYNONYM HYPEREDGE CARDINALITY DISTRIBUTION** Figure 8.12 illustrates the distribution of synonyms per hyperedge. As we can see, most *synonym* hyperedges either contain two or three terms, while less than 100 hyperedges contain more than five synonyms. Most synonymy relations are ternary and, while there is not enough data to conclude it, the overall behavior approximates a power law.

**AVERAGE HYPEREDGE CARDINALITY OVER TIME** Consistent with the fact that most synsets introduced as undirected hyperedges have a low cardinality (two or three elements), the average hyperedge cardinality over time is overall lower than the base model. This is visible when comparing Figure 8.14 with Figure 8.4. Additionally, the behavior also changed from a fast growth and convergence behavior, in the base model, to a consistent sub-linear growth behavior. While convergence is not immediately clear in the synonyms model, the trend does point to such behavior.

**AVERAGE ESTIMATED DIAMETER AND AVERAGE PATH LENGTH OVER TIME** With synonymy relations, both the average path length and the diameter start at a lower value than the base model, for only one document. Apart from the initial values, when comparing Figure 8.15 with Figure 8.5, we find a similar behavior, although the average path length decreases from 8.37, in the base model, to 7.53, in the synonyms model, when comparing a representation of the whole collection (cf. Tables 8.1 and 8.2). Despite the similar behavior, a unitary difference is quite significative in a network (e.g., in a social network like Facebook, the average path length is 4.74 [349], while in the original small-world study by Milgram [350, 351] the average path length was 6.2).

**TEMPORAL STATISTICS OF RUN TIMES** Finally, Figure 8.16 illustrates the synonyms model run times of the following operations for an increasing number of documents: index creation (8.16a); the computation of the global statistics (8.16b), also shown in Table 8.2; the computation of all node degrees (8.16c); and the computation of all hyperedge cardinalities (8.16d). As we can see, similarly to what




**(a)** Index creation.

**(b)** Global statistics computation.

**(c)** Node degrees computation.

**(d)** Hyperedge cardinalities computation.

**Figure 8.16:** Synonyms model run time statistics.

happened for the base model, the most significant increase in run time happens around 1,000 documents, with the exception of the global statistics computation, which shows an increased run time for the first added documents. We predict that the same caching mechanisms described for the base model are responsible for this anomaly. In Figure 8.16c, we also find a slight decrease in run time from 5,000 to 8,000 documents, which we do not find significant, as it was perhaps due to temporary load on the virtual machine. Indexing time took 1m13s for 1,000 documents and 4m22s for a maximum of 8,000 documents. The computation of global statistics took 17m07s for 1,000 documents and 39m13s for a maximum of 8,000 documents. Node degrees were computed in 4m11s for 1,000 documents, taking 19m03s at most, while hyperedge cardinalities were computed in only 20s for 1,000 documents, taking 44s at most, and maintaining the top rank in the most efficient statistic to compute, when compared to the base model.

### 8.4.2 Contextual similarity

Another way that we extended the base model was by using the contextual similarity between terms, as established based on the k-nearest neighbors according to word embeddings. For this particular analysis, word embeddings were obtained through word2vec, trained on a larger subset of the INEX 2009 Wikipedia collection, built from the documents mentioned in the relevance judgments for all 52 topics. The extracted vectors were of size 100, using sliding windows of 5 words to establish context, and ignoring words that appeared only once. Only the two nearest neighbors, with a similarity above 0.5 were considered to build the similarity graph. Contextual similarity hyperedges were then derived from this graph by iterating over each term and building sets that included the original term as well as incoming and outgoing terms.

Table 8.3 shows the global statistics for the context model. As we can see, the number of terms significantly increased from 323,672 (cf. Table 8.1) to 413,527. This means that 89,855 contextually similar terms that did not originally belong to the collection were added — they were however a part of the larger 52 topics collection, otherwise no new terms would have been added. The number of undirected hyperedges also increased significantly, with 157,217 new context relations. The average degree also increased from 0.83 to 1.18, with the average clustering coefficient remaining stable and the density decreasing from 3.88e−06 to 2.75e−06. The

**Table 8.3**: Global statistics for the contextual similarity model.

| Statistic | Value |
|---|---|
| Nodes | 697,068 |
| *term* | 413,527 |
| *entity* | 283,541 |

| Statistic | Value |
|---|---|
| Hyperedges | 410,371 |
| Undirected | 172,155 |
| *document* | 7,484 |
| *related_to* | 7,454 |
| *context* | 157,217 |
| Directed | 238,216 |
| *contained_in* | 238,216 |

| Statistic | Value |
|---|---|
| Avg. Degree | 1.1774 |
| Avg. Clustering Coefficient | 0.1423 |
| Avg. Path Length | 1.9333 |
| Diameter | 3 |
| Density | 2.75e-06 |

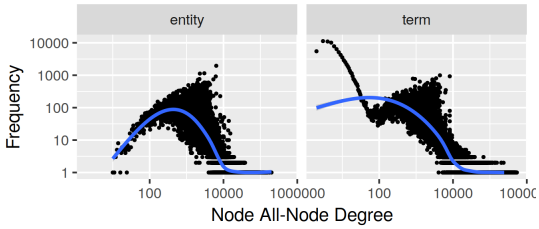



**Figure 8.17**: Node-based degree distribution for the context model (log-log scale).

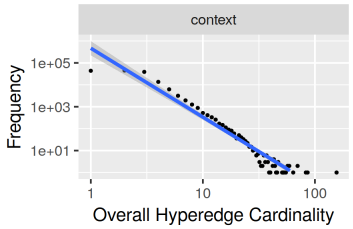



**Figure 8.18**: Context hyperedge cardinality distribution (log-log scale).

diameter significantly decreased from 17 to 3, as did the average path length, which decreased from 8.37 to 1.93, strongly approximating nodes through the relation of contextual similarity. This is an indicator of the impact of using word embeddings to establish new bridges between documents, although we need to assess whether retrieval effectiveness will be affected by context as a kind of noise introduced in the process rather than a good discriminative feature. We found 42,145 new paths created by this extension, resulting in 23.03 documents linked on average per context. Notice that, although synonyms established a lower number of bridges, they also connected a higher number of documents on average (2.83× more than context). Only by studying retrieval effectiveness are we able to assess which characteristic translates into a better performance in the model. Besides global statistics, we also identified four interesting changes or new characteristics when compared to the base model:

- Term node degree distribution;
- Context hyperedge cardinality distribution;
- Average hyperedge cardinality over time;
- Average estimated diameter and average path length over time;

**TERM NODE DEGREE DISTRIBUTION** Figure 8.17 illustrates the node-based node degree distribution for entity and term nodes in the hypergraph-of-entity with the context extension. The behavior for entity nodes is similar to the base model and to the synonyms model. However, like in the synonyms model, term nodes show a combination of a power law like behavior for the lower degrees, with a log-linear behavior for the remaining degrees. Given the higher number of terms introduced through contextual similarity, we also find a distribution plot that is visually denser.

**CONTEXT HYPEREDGE CARDINALITY DISTRIBUTION** Figure 8.18 illustrates the distribution of terms per *context* hyperedge. As we can see, the behavior approximates a power law, with only a few *context* hyperedges containing around 50 nodes and one of them even reaching 156 nodes.

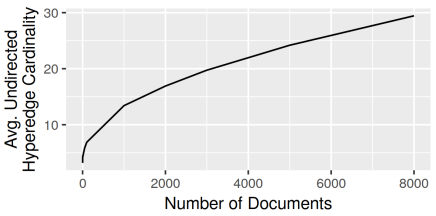



**Figure 8.19:** Average hyperedge cardinality over time for the context model.

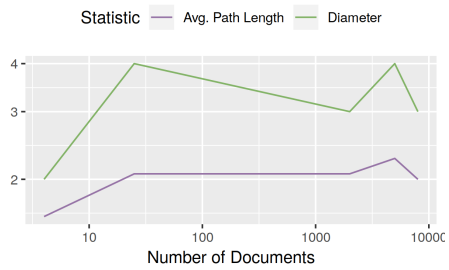
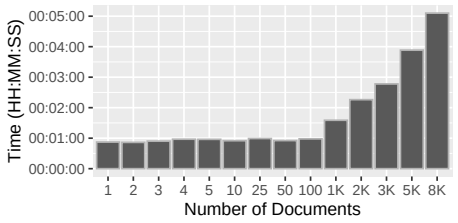
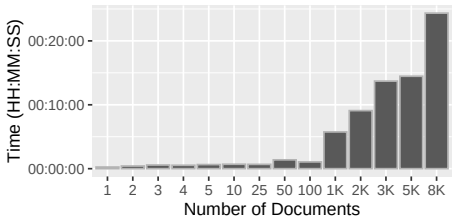


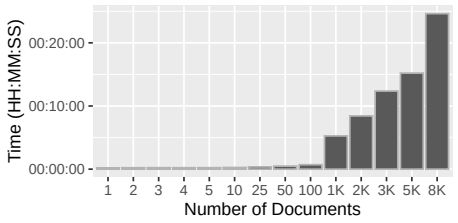

**Figure 8.20:** Average estimated diameter and average shortest path over time for the context model.



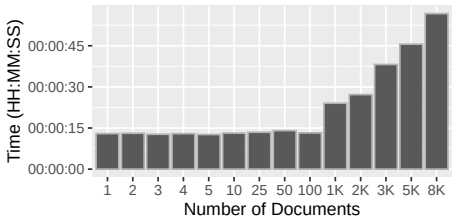

**(a)** Index creation.




**(b)** Global statistics computation.




**(c)** Node degrees computation.




**(d)** Hyperedge cardinalities computation.

**Figure 8.21:** Contextual similarity model run time statistics.

**AVERAGE HYPEREDGE CARDINALITY OVER TIME** Given the high number of introduced *context* hyperedges, most of them with a low cardinality, the average hyperedge cardinality was driven down, as we can see in Figure 8.19. In a similar way to the *synonym* hyperedges, the behavior also changed from a fast growth and convergence behavior, in the base model, to a consistent sub-linear growth behavior.

**AVERAGE ESTIMATED DIAMETER AND AVERAGE PATH LENGTH OVER TIME** Perhaps one of the most interesting results of this analysis is the impact of index extensions in the diameter and average path length. This is particularly visible with the context extension — the diameter decreased from 17, in the base and similarity models, to only 3, in the context model. A similar behavior was identified for the average path length that decreased from 8.33 in the base model and 7.53 in the synonyms model, to only 1.93 in the context model. This behavior over time is seen in Figure 8.20, where, contrary to the base and synonyms model, we can find shorter geodesics immediately for a low number of documents. As an increasing part of the collection is considered, the length of the geodesics increase. This might be correlated with an increasing diversity of topics, thus being indicative of the discriminative power of the context extension, an aspect that should be further investigated in the future.

**TEMPORAL STATISTICS OF RUN TIMES** Finally, Figure 8.21 illustrates the contextual similarity model run times of the following operations for an increasing number

**Table 8.4:** Global statistics for the TF-bins model (bins=2 and bins=10).

| Statistic | Bins | |
|---|---|---|
| | 2 | 10 |
| Nodes | 607,213 | 607,213 |
| *term* | 323,672 | 323,672 |
| *entity* | 283,541 | 283,541 |

| Statistic | Bins | |
|---|---|---|
| | 2 | 10 |
| Hyperedges | 268,100 | 281,642 |
| Undirected | 29,884 | 43,426 |
| *document* | 7,484 | 7,484 |
| *related_to* | 7,454 | 7,454 |
| *tf_bin* | 14,946 | 28,488 |
| Directed | 238,216 | 238,216 |
| *contained_in* | 238,216 | 238,216 |

| Statistic | Bins | |
|---|---|---|
| | 2 | 10 |
| Avg. Degree | 0.8831 | 0.9277 |
| Avg. Cl. Coef. | 0.1021 | 0.1014 |
| Avg. Path Len. | 6.8333 | 6.9000 |
| Diameter | 13 | 14 |
| Density | 7.58e-06 | 7.86e-06 |

of documents: index creation (8.21a); the computation of the global statistics (8.21b), also shown in Table 8.3; the computation of all node degrees (8.21c); and the computation of all hyperedge cardinalities (8.21d). As we can see, similarly to what happened for the base model, the most significant increase in run time happens around 1,000 documents. When compared to the base model and the synonyms model, the global statistics computation does not show an increased run time for the first added documents. This further supports the hypothesis of this being an anomaly that happened due to initial caching or load issue, particularly since the synonyms model is quite similar, structurally, to the context model. Indexing time took 1m35s for 1,000 documents and 5m05s for a maximum of 8,000 documents. The computation of global statistics took 5m44s for 1,000 documents and 24m20s for a maximum of 8,000 documents. Node degrees were computed in 5m15s for 1,000 documents, taking 24m37s at most, while hyperedge cardinalities were computed in only 24s for 1,000 documents, taking 56s at most, making it the most efficient statistic to compute, and maintaining the top rank in the most efficient statistic to compute, when compared to the base model and the synonyms model.

### 8.4.3 Term frequency bins

In this section, we analyze the TF-bins extension, which is based on the discretization of the term frequency per document. This way, term frequency can be added to the hypergraph-of-entity, while having a low impact in scalability (i.e., we remain focused on forming groups of nodes to minimize the space complexity of the representation model).

Table 8.4 shows the global statistics for the TF-bins model. As we can see, the number of nodes is the same as the original model, also remaining unchanged with the number of bins. The number of undirected hyperedges increased from 14,938 to 29,884 for two TF-bins, or to 43,426 with ten bins. The average degree slightly increased from 0.83 to 0.88 for two TF-bins per document, and then to 0.93 for ten TF-bins, with the average clustering coefficient remaining stable and the density increasing from 3.88e−06 to 7.58e−06 for two TF-bins, and then again slightly to 7.86e−06 for ten TF-bins. The diameter decreased from 17 to 13 for two TF-bins, and 14 for ten TF-bins, as did the average path length, which decreased from 8.37 to 6.83 and 6.90 for two and ten TF-bins, respectively. When considering two TF-bins, we found 156,200 new paths created by this extension, resulting in 30.64 documents linked on average per TF-bin. When the number of bins increased to ten, the number of new paths decreased to 153,979, but the average number of documents linked per TF-bin increased to 37.99. Besides global statistics, we also identified seven interesting changes or new characteristics when compared to the base model:

- TF-bin hyperedge cardinality distribution per number of bins;
- Number of undirected hyperedges per number of bins;
- TF-bin hyperedges per number of bins;

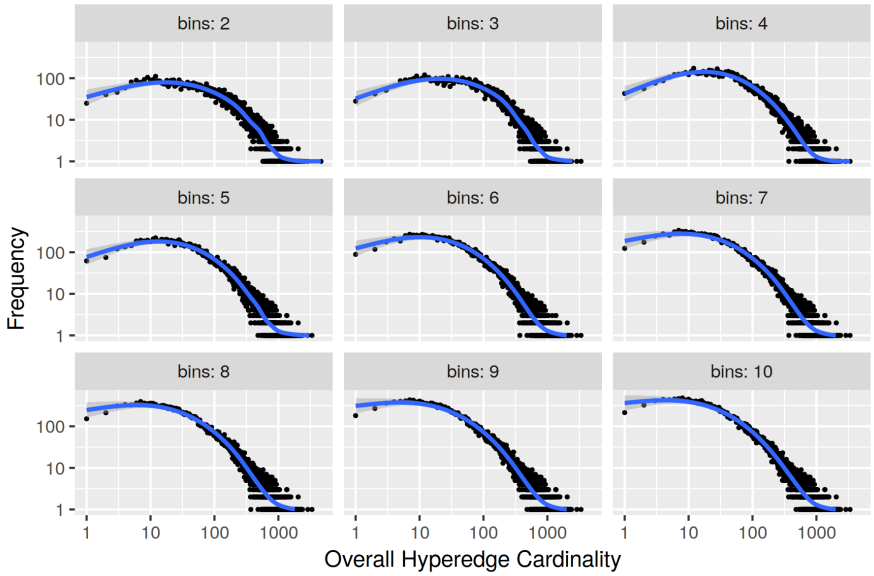
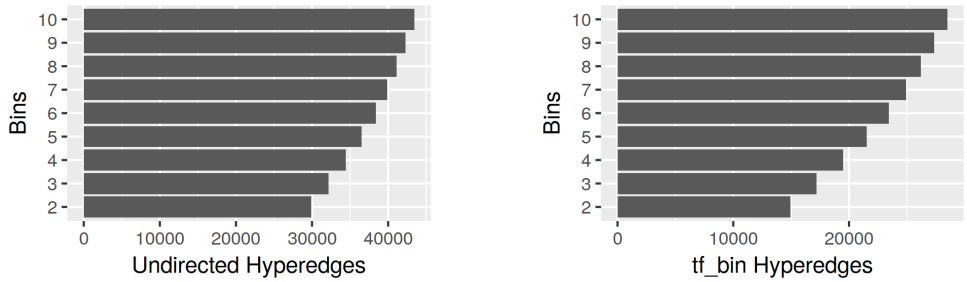



**Figure 8.22:** TF-bin hyperedge cardinality distribution (log-log scale).

**Figure 8.23:** Number of hyperedges, per number of bins, for the TF-bins model.




**(a)** Undirected hyperedges.

**(b)** TF-bin hyperedges.

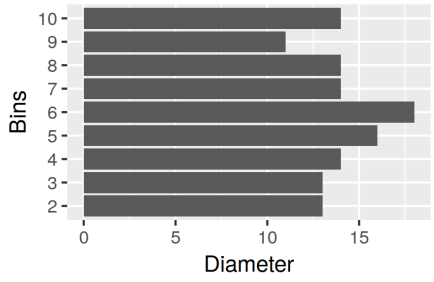
- Diameter and average path length per number of bins;
- Average hyperedge cardinality over time per number of bins;
- Average density over time per number of bins.
- Average estimated diameter and average path length over time per number of bins;

Notice that, contrary to the synonyms and context extensions, the TF-bins extension did not affect the behavior of term node degree distribution, since it does not introduce external terms to the collection.

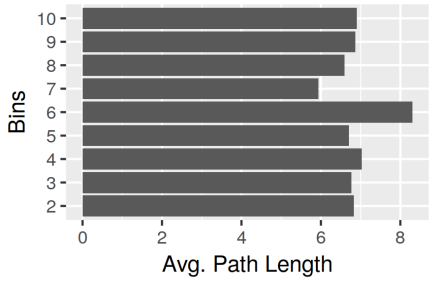
**TF-BIN HYPEREDGE CARDINALITY DISTRIBUTION** Figure 8.22 illustrates the cardinality distribution of *tf_bin* hyperedges, for different numbers of bins. The behavior is similar to the *related_to* hyperedges, however, as the number of bins increases, lower values of cardinality become more frequent and the behavior starts tending towards a power law.

**NUMBER OF HYPEREDGES PER NUMBER OF BINS** As expected, in Figure 8.24a, we find a growth in the number of undirected hyperedges, from 29,884, for two bins, to 43,426, for ten bins. The same happens for the *tf_bin* hyperedges (Figure 8.24b), which are responsible for propelling such growth. The amount of hyperedges generated by increased TF-bins will eventually converge, since there is a limited number

**Figure 8.24:** Geodesic-based metrics, per number of bins, for the TF-bins model.




**(a)** Diameter.




**(b)** Average path length.

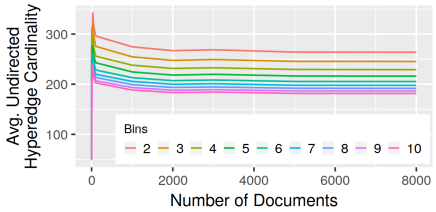



**Figure 8.25:** Average hyperedge cardinality over time, per number of bins, for the TF-bins model.

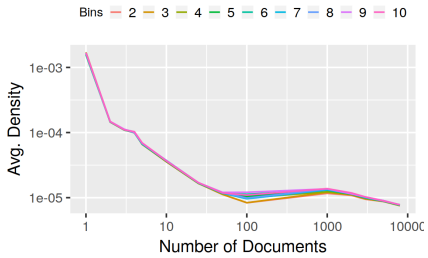



**Figure 8.26:** Average density over time, per number of bins, for the TF-bins model.

of terms per document to segment. However, for this collection, it is clear that the number of TF-bins can range from two to ten, while always generating new hyperedges, increasing the granularity at which term frequency contributes to the model.

**DIAMETER AND AVERAGE PATH LENGTH PER NUMBER OF BINS** As show in Figure 8.24, both the diameter and the average path length, which correspond to the maximum and average geodesic distances in the hypergraph, show a high variability with the number of bins. In particular, the diameter and average path length both reach their maximum values of 18 and 8.30 when using 6 TF-bins. The minimum diameter of 11 is reached when using 9 TF-bins, while the minimum average path length of 5.93 is reached when using 7 TF-bins. This suggests that the number of bins might influence retrieval effectiveness, if varying the diameter and the average path length also affects performance directly.

**AVERAGE HYPEREDGE CARDINALITY OVER TIME** Figure 8.26 shows the evolution of the average hyperedge cardinality for different numbers of bins. The behavior is similar to the base model (cf. Figure 8.4), which is equivalent to having one TF-bin. As the number of TF-bins increases, the overall average hyperedge cardinality decreases, which is the expected behavior. This is less visible as the number of bins reaches a higher value, at which point the overall cardinality is less affected, showing a progressively lower decreasing behavior. While the number of TF-bins affects this characteristic of the hypergraph, the overall behavior is maintained.

**AVERAGE DENSITY OVER TIME** The average density shown in Figure 8.26 follows a similar behavior to the base model (cf. Figure 8.7), regardless of the number of TF-bins. However, there is a small variation for the interval of approximately 100 to 1,000 documents, after which it is once again reduced to the same value for the different numbers of TF-bins. It is perhaps the diversity in term frequency

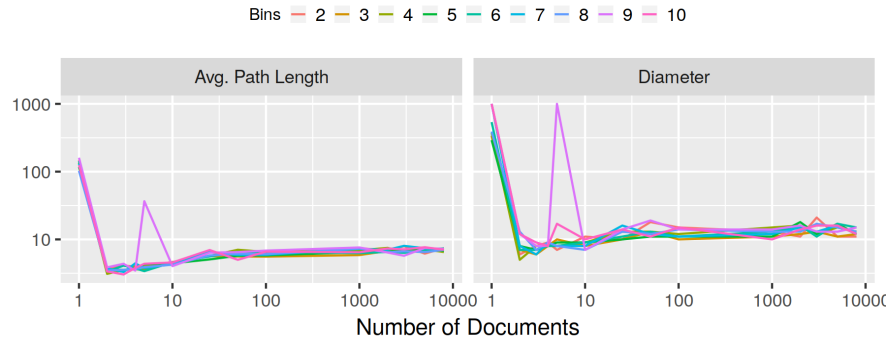
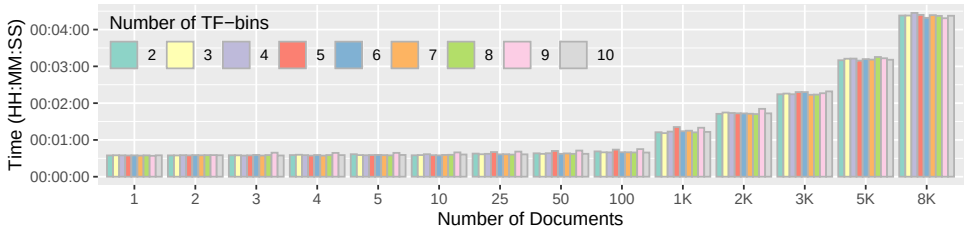
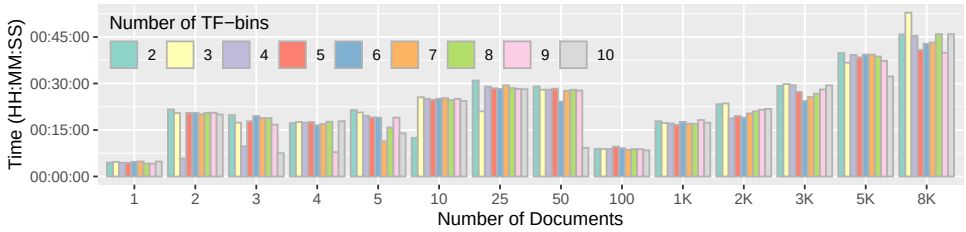



**Figure 8.27:** Average estimated diameter and average shortest path over time, per number of bins, for the TF-bins model.




**(a)** Index creation.



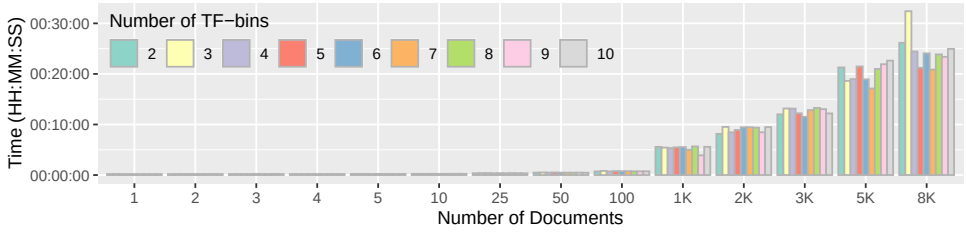

**(b)** Global statistics computation.

**Figure 8.28:** TF-bins models run time statistics (part 1).

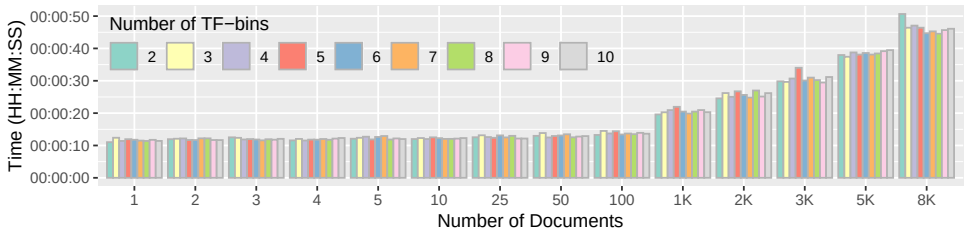
introduced for documents in this interval that promotes such a difference. This would explain the creation of a higher number of *tf_bin* hyperedges, without empty TF intervals (e.g., $]2, 2]$).

**AVERAGE ESTIMATED DIAMETER AND AVERAGE SHORTEST PATH OVER TIME** Figure 8.27 shows the evolution of the diameter and average path length, over an increasing number of documents and TF-bins. Apart from both metrics reaching higher values for a single document as well as for five TF-bins, the behavior is similar to the base model (cf. Figure 8.5).

**TEMPORAL STATISTICS OF RUN TIMES** Finally, Figures 8.28 and 8.29 illustrate the TF-bins model run times of the following operations for an increasing number of documents: index creation (8.28a); the computation of the global statistics (8.28b), also shown in Table 8.4; the computation of all node degrees (8.29a); and the computation of all hyperedge cardinalities (8.29b). As we can see, similarly to what happened for the base model and the synonyms model, the most significant increase in run time happens around 1,000 documents, with the exception of the global statistics computation, which shows an increased run time for the first added documents. Indexing time took 1m11s for 1,000 documents and 4m27s for a maximum of 8,000 documents. The computation of global statistics took 16m38s for 1,000 documents




**(a)** Node degrees computation.




**(b)** Hyperedge cardinalities computation.

**Figure 8.29:** TF-bins models run time statistics (part 2).

and 52m50s for a maximum of 8,000 documents. Node degrees were computed in 3m54s for 1,000 documents, taking 32m23 at most, while hyperedge cardinalities were computed in only 19s for 1,000 documents, taking 50s at most, making it the most efficient statistic to compute, maintaining the top rank in the most efficient statistic to compute, in line with the other studied models models.

## 8.5 AN APPLICATION TO INFORMATION RETRIEVAL

So far, we have analyzed the structural impact of different index extensions in regard to the characteristics of the hypergraph. However, there is little value in understanding the behavior of structural features without the context of its application, which in this case is in the area of information retrieval [303]. Thus, we assess the effectiveness of each model, with different extensions and parameter configurations, through a classical information retrieval evaluation process, based on the 10 topic subset of the INEX 2009 Wikipedia collection (Section 10.2.2).

We launched three evaluation runs per index configuration, i.e., for different versions of the HGoE (HyperGraph-of-Entity) representation model based on different extensions. We relied on the RWS function, experimenting with different random walk lengths $\ell \in \{1,2,3\}$, and a fixed configuration for the remaining parameters: $r = 10{,}000$, *expansion* disabled (i.e., without seed node selection [303, §4.2.1]), and *weight*s enabled (i.e., considering *tf_bin* hyperedge weights, the only available weights in the indexes).

Table 8.5 shows the mean average precision (MAP), normalized discounted cumulative gain for the top 10 results (NDCG@10), and precision for the top 10 results (P@10), computed for the relevance judgments provided by the INEX 2010 Ad Hoc track [248]. As we can see by analyzing the maximum values per column (in bold), the TF-bin models were able to obtain significantly better results overall, when compared to the base model, the synonyms model, and the context model. None of the HGoE models is yet able to outperform the baselines, although TF-bins are able to approximate TF-IDF in regard to NDCG@10 and P@10. The hypergraph-based models need to be reiterated over and improved. Herein lies the usefulness of computing the properties of the hypergraph structures and analyzing the hypergraph-of-entity. While there is no clear pattern of effectiveness correlated with the number of bins, if we consider the NDCG@10 scores, the best model for $\ell = 1$ is TF-bins$_2$,

**Table 8.5:** Evaluating the different models in the ad hoc document retrieval task.

| Model | MAP | NDCG@10 | P@10 | MAP | NDCG@10 | P@10 | MAP | NDCG@10 | P@10 |
|---|---|---|---|---|---|---|---|---|---|
| Lucene TF-IDF | 0.2160 | 0.2667 | 0.2800 | 0.2160 | 0.2667 | 0.2800 | 0.2160 | 0.2667 | 0.2800 |
| Lucene BM25 | 0.3412 | 0.5479 | 0.4900 | 0.3412 | 0.5479 | 0.4900 | 0.3412 | 0.5479 | 0.4900 |
| HGoE RWS | | $\ell = 1$ | | | $\ell = 2$ | | | $\ell = 3$ | |
| Base model | 0.0046 | 0.0799 | 0.0400 | 0.0039 | 0.0718 | 0.0400 | 0.0028 | 0.0576 | 0.0400 |
| Synonyms | 0.0013 | 0.0440 | 0.0200 | 0.0024 | 0.0799 | 0.0400 | 0.0023 | 0.0718 | 0.0400 |
| Context | 0.0000 | 0.0000 | 0.0000 | 0.0010 | 0.0220 | 0.0100 | 0.0010 | 0.0220 | 0.0100 |
| TF-bins$_2$ | **0.1082** | **0.2443** | 0.2100 | 0.1025 | 0.1730 | **0.2000** | 0.0918 | 0.1302 | 0.1400 |
| TF-bins$_3$ | 0.0911 | 0.2004 | 0.2200 | 0.0989 | 0.0954 | 0.1200 | 0.0868 | 0.0751 | 0.1000 |
| TF-bins$_4$ | 0.0957 | 0.1969 | 0.2000 | 0.1107 | **0.2007** | 0.1900 | 0.0928 | 0.1669 | 0.1700 |
| TF-bins$_5$ | 0.1049 | 0.2355 | 0.2400 | 0.1050 | 0.1364 | 0.1400 | 0.0954 | 0.1121 | 0.1400 |
| TF-bins$_6$ | 0.1057 | 0.2405 | **0.2600** | 0.1108 | 0.1906 | **0.2000** | 0.1022 | **0.1792** | **0.1900** |
| TF-bins$_7$ | 0.1000 | 0.2212 | 0.2500 | 0.1072 | 0.1255 | 0.1200 | 0.0939 | 0.0934 | 0.1000 |
| TF-bins$_8$ | 0.0894 | 0.2131 | 0.2100 | 0.1078 | 0.0988 | 0.1100 | 0.0966 | 0.0641 | 0.0800 |
| TF-bins$_9$ | 0.0954 | 0.1494 | 0.1500 | 0.1107 | 0.1402 | 0.1500 | 0.0958 | 0.1069 | 0.1200 |
| TF-bins$_{10}$ | 0.1062 | 0.2127 | 0.2200 | **0.1133** | 0.1436 | 0.1600 | **0.1079** | 0.1143 | 0.1300 |

**Table 8.6:** Comparing the global statistics for the different models.

| Model | Nodes | Hyperedges | Degree | Cl. Coef. | Avg. Path Len. | Diam. | Density |
|---|---|---|---|---|---|---|---|
| Base model | 607,213 | 253,154 | 0.8338 | 0.1148 | 8.3667 | 17 | 3.88e-06 |
| Synonyms | 610,212 | 263,804 | 0.8646 | 0.1168 | 7.5333 | 17 | 3.88e-06 |
| Context | 697,068 | 410,371 | 1.1774 | 0.1423 | 1.9333 | 3 | 2.75e-06 |
| TF-bins$_2$ | 607,213 | 268,100 | 0.8831 | 0.1021 | 6.8333 | 13 | 7.58e-06 |
| TF-bins$_3$ | 607,213 | 270,359 | 0.8905 | 0.1011 | 6.7667 | 13 | 7.65e-06 |
| TF-bins$_4$ | 607,213 | 272,649 | 0.8980 | 0.0999 | 7.0333 | 14 | 7.60e-06 |
| TF-bins$_5$ | 607,213 | 274,698 | 0.9048 | 0.0996 | 6.7000 | 16 | 7.73e-06 |
| TF-bins$_6$ | 607,213 | 276,615 | 0.9111 | 0.1029 | 8.3000 | 18 | 7.69e-06 |
| TF-bins$_7$ | 607,213 | 278,087 | 0.9159 | 0.1010 | 5.9333 | 14 | 7.82e-06 |
| TF-bins$_8$ | 607,213 | 279,356 | 0.9201 | 0.1034 | 6.6000 | 14 | 7.83e-06 |
| TF-bins$_9$ | 607,213 | 280,524 | 0.9240 | 0.0994 | 6.8667 | 11 | 7.84e-06 |
| TF-bins$_{10}$ | 607,213 | 281,642 | 0.9277 | 0.1014 | 6.9000 | 14 | 7.86e-06 |

the best model for $\ell = 2$ is TF-bins$_4$, and the best model for $\ell = 3$ is TF-bins$_6$. This might indicate that a higher number of bins works best with a longer random walk length. However, there is no concordance to support this hypothesis when looking at the MAP and P@10 metrics, thus further investigation is required.

In order to better understand whether there is a direct relation between any of the computed structural features of the hypergraph and the effectiveness of the retrieval model, we first summarize the structural features for each model in Table 8.6. By comparing each feature with the evaluation metrics from Table 8.5, we are able to find some indicators of (in)effectiveness in a graph-based retrieval model. According to Table 8.5, context was the worst performing model, over all values of $\ell$. The context model also has the highest average degree and clustering coefficient, as well as the lowest average path length and diameter (cf. Table 8.6). This indicates that a higher local connectivity and an overall lower distance between nodes might not beneficial for retrieval effectiveness. We also observe that the TF-bin models, which have the best performance, also have a lower clustering coefficient than the base, synonyms and context models, ranging between 0.0994 and 0.1034.

We also studied the structural impact of each extension, through the relative change to individual features, in comparison to the base model. Figure 8.30 shows a heatmap based on the change percentages in regard to the base model, which, by definition, has a 0% change over all features, in comparison to itself. As we can see, the context model suffered the most evident overall change, with a $-467\%$ change in diameter, and a $-333\%$ change in average path length. This model is of particular interest, as it resulted in the worst retrieval performance, when compared to the remaining models. Interestingly, this is also visible in its structural features. The clustering coefficient for the context model also suffered a substantial increase in relation to the base model, with a change of 19%, as did the degree, with a

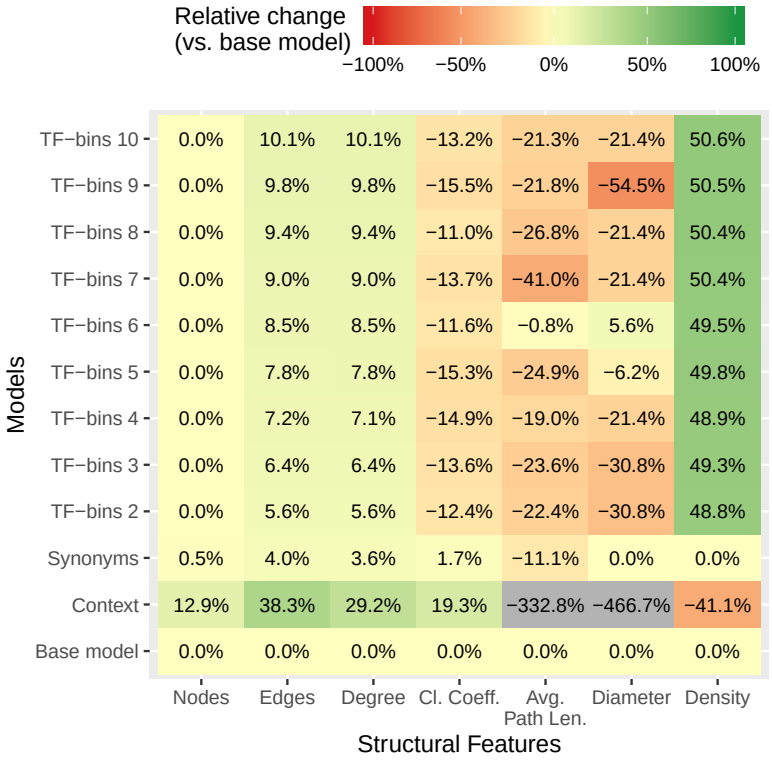



**Figure 8.30:** Relative change of structural features when compared to the base model.

**Table 8.7:** Spearman's ρ between evaluation metrics and structural features.

| | | Nodes | Hyperedges | Degree | Cl. Coef. | Avg. Path Len. | Diam. | Density |
|---|---|---|---|---|---|---|---|---|
| $\ell = 1$ | **MAP** | -0.6504 | 0.0559 | 0.0559 | -0.5245 | 0.0979 | 0.1000 | 0.5009 |
| | **NDCG@10** | -0.6504 | -0.0350 | -0.0350 | -0.3636 | -0.1119 | 0.2000 | 0.4308 |
| | **P@10** | -0.6527 | 0.1018 | 0.1018 | -0.4667 | -0.0982 | 0.3047 | 0.5800 |
| $\ell = 2$ | **MAP** | -0.6516 | 0.4098 | 0.4098 | -0.5464 | 0.2172 | 0.1449 | 0.8035 |
| | **NDCG@10** | -0.5913 | 0.0699 | 0.0699 | -0.5804 | 0.2797 | 0.1036 | 0.4448 |
| | **P@10** | -0.6242 | 0.0035 | 0.0035 | -0.5519 | 0.2882 | 0.0593 | 0.4049 |
| $\ell = 3$ | **MAP** | -0.6504 | 0.4615 | 0.4615 | -0.4685 | 0.0699 | 0.1965 | 0.8932 |
| | **NDCG@10** | -0.5322 | -0.0280 | -0.0280 | -0.5524 | 0.3357 | 0.2000 | 0.3573 |
| | **P@10** | -0.6242 | -0.0211 | -0.0211 | -0.6151 | 0.2707 | 0.1993 | 0.3873 |

change of 29%. When looking at the density for all models, there was no change for the synonyms model, but there was a positive change, rounding 50% (in green), for the TF-bins models, and there was a negative change of −41% for the context model. The number of nodes suffered no change for the TF-bins models, but there a slight increase for synonyms (as new terms from synsets were added), and a more significative increase for the context model. The number of edges suffered a consistently larger increase for TF-bins models, as the number of bins increased, with the synonyms model showing a slight increase, and the context model once again showing a more significative increase.

### 8.5.1 Correlating evaluation metrics and structural features

In Table 8.7 we further organize this approach, by comparing the evaluation results of each metric with the values of each structural feature. By using Spearman's rank correlation coefficient (ρ), we can verify whether the retrieval model's performance ranking given by the evaluation metrics (our ground truth) can compare with the ranking given by any of the structural features, as computed for each model. Let

**Table 8.8:** Indicators of graph-based retrieval model performance.

| Ranking indicators | | Anomaly indicators | |
|---|---|---|---|
| Cl. Coef. | Ascending order ~ 50% correlated with retrieval performance. | Degree | Abnormally high values ($> \mu + 2\sigma$) indicate a low performing model. |
| Density | Descending order ~ 50% correlated with retrieval performance. | Diameter | Abnormally low values ($< \mu - 2\sigma$) indicate a low performing model. |

us first follow up with the indicators we put forth in the manual comparison of the two tables.

We proposed that a high average degree and clustering coefficient would result in a low MAP, NDCG@10 and P@10, which does not necessarily mean that either feature is a good overall discriminator of model performance. In fact, the average degree does not show correlation consistency among the different evaluation metrics and parameter configurations. On the other hand, the clustering coefficient is negatively correlated with each evaluation metric over the different random walk length parameter configurations, ranging between $-0.61$ and $-0.36$. This makes the clustering coefficient a weak, but consistent indicator of the performance of graph-based retrieval models (i.e., higher values of the clustering coefficient indicate a low retrieval effectiveness). Absolute correlation is not particularly high, since retrieval performance does not solely depend on the structure of the graph, but also on the semantics of the representation model.

We also proposed that a low average path length and diameter would be indicative of low model performance. While the average path length and diameter correlations with the evaluation metrics are mostly positive, these are not sufficiently consistent to be considered good global indicators of performance. There are, however, special cases when the average path length serves as a slight indicator of performance, namely for $\ell > 1$ and for the top 10 results. For $\ell = 1$, there is a slight negative correlation that could be explained by the fact that this model only relies on the immediate neighborhood within the hypergraph and does not depend on short paths for connectivity. The diameter, on the other side, always shows a positive correlation with the evaluation metrics, but its absolute value is overall low and inconsistent for it to provide a good discriminative indicator of retrieval performance.

With a similar behavior to the clustering coefficient, but with an inverse sign, the density was overlooked as a good indicator of model performance. In particular, the worst performing model (context model) also has the lowest density of $2.75e{-}06$, followed by the base model and the synonyms model, tied at a density of $3.88e{-}06$, and then by the TF-bin models, with densities ranging from $7.58e{-}06$ to $7.86e{-}06$. While the density is a good discriminative of graph-based retrieval models, its granularity is low, only properly distinguishing between models with an obvious difference in performance.

### 8.5.2 Design rules for modifying or extending the model

After the analysis of the impact of structural features in the performance of the retrieval models, we reflect on the implications of our findings. We use these findings to prepare a set of rules that serve as indicators or as a guide for the continued redesign of the hypergraph-of-entity. In particular, the guidelines we propose should be helpful in the process of comparing different versions based on modifications or extensions to our model. We propose two classes of indicators:

**RANKING INDICATORS** Structural features that can be used to rank different graph-based models in regard to their predicted retrieval performance.

**ANOMALY INDICATORS** Structural features that cannot be used to rank graph-based models based on retrieval performance, but can, however, be useful for identifying anomalous models with a high chance of a low performance.

Table 8.8 shows the identified ranking and anomaly indicators according to the analysis carried at the beginning of this section. The clustering coefficient and the density were both identified as ranking indicators with an approximate certainty rate of 50%, based on an ascending and descending order, respectively. The degree and diameter were identified as anomaly indicators, with the degree being used to identify abnormally high values, for example larger than two standard deviations ($2\sigma$) above the mean ($\mu$), and the diameter being used to identify abnormally low values, for example less than two standard deviations below the mean.

## SUMMARY

In this chapter, we characterized the hypergraph-of-entity representation model, based on the structural features of the hypergraph. We analyzed the node degree distributions, based on nodes and hyperedges, and the hyperedge cardinality distributions, illustrating their distinctive behavior. We found that hyperedge-based node degrees are distributed as a power law, while node-based node degrees and hyperedge cardinalities are log-normally distributed. We also analyzed the temporal behavior, as documents were added to the index, studying average node degree and hyperedge cardinality, estimated average path length, diameter and clustering coefficient, as well as density and space usage requirements. We found that most statistics tend to converge after an initial period of accentuated growth in the number of documents.

We then expanded on the characterization work by analyzing different model extensions based on synonymy, contextual similarity, and the newly introduced concept of TF-bins, and we also measured the run time of several operations, like indexing and the computation of hypergraph properties. Our contributions included the application of two strategies for the approximation of statistics based on the shortest distance, as well as the clustering coefficient. Additionally, we proposed a simple approach for computing the density of a general mixed hypergraph, based on an induced bipartite mixed graph.

Finally, we focused on the application of this characterization work, which can be used to inform the design of graph-based representation models for information retrieval. In particular, we studied the change in structural features, when compared to the base model, as well as the correlations between retrieval effectiveness metrics (MAP, NDCG@10, P@10) and structural features (e.g., average degree, clustering coefficient). While structural features rarely presented a higher than 50% absolute correlation with any of the evaluation metrics, we identified some of them as useful indicators for ranking the retrieval models according to their effectiveness, or for identifying anomalies that lead to low effectiveness.

More importantly, we have provided an analysis framework for hypergraphs that can easily be implemented and applied to both small and large-scale hypergraphs. We have also provided a characterization based on this framework, illustrating the behavior of several statistics, for instance showing that, while the degree distribution based on hyperedges still follows a power law, like in real-world networks represented as graphs, the degree distribution based on nodes instead approximates a log-normal distribution. During the development of this work, we have also found that:

- Few attention has been given to hypergraph characterization in the real-world;
- The community is still lacking in tools to analyze hypergraphs:
  - There is no *de facto* library for hypergraph analysis;
  - Few file formats support hypergraphs, namely with directed hyperedges.
- Polyadism introduces additional complexity and calls for novel metrics that take the information within collective relations into account.

# 9 EVALUATING THE HYPERGRAPH-OF-ENTITY GENERAL REPRESENTATION AND RETRIEVAL MODEL

## Contents



In Chapter 7, the hypergraph-of-entity was conceptualized and described as a general model for entity-oriented search. In Chapter 8, we characterized the structure of the representation model. In this chapter, we focus on evaluating the retrieval effectiveness of the hypergraph-of-entity, applying the random walk score to multiple entity-oriented search tasks, while testing a wide range of configurations both for the representation model and the ranking function. Although our goal is to improve effectiveness, or a the very least ensure a minimum level of effectiveness, we also measure the efficiency of each run, in order to understand the overall impact and tradeoff of each configuration.

We begin by measuring the performance over a single retrieval task, ad hoc document retrieval (leveraging entities), in order to focus on different versions of the hypergraph-of-entity representation model. We then expand this line of research to other retrieval tasks, also evaluating ad hoc entity retrieval, and entity list completion. We do this over a common index data structure and using the random walk score as a universal ranking function. Since the different tasks cannot be compared among each other, we also attempt to scale the model, so that it can index the complete INEX 2009, as opposed to just the subsets that we rely on for the first experiments. We do this by indexing the top keywords for each document, reducing complexity by partially lowering the number of nodes and, indirectly, the number of hyperedges linking terms to entities. This enables us to compare the effectiveness of the hypergraph-of-entity with the results obtained by the participants

of the INEX tracks for the considered tasks. Despite its low performance, we find this to be a viable approach that is, to our knowledge, the first attempt at a general representation model that supports a universal ranking function for entity-oriented search tasks.

The structure of this chapter is organized as follows:

- **Section 9.1** describes several experiments over the INEX 2009 Wikipedia subsets, mainly over the INEX 2009 52T-NL, which includes all topics but only the documents present in the relevance judgments. We provide statistics about the created hypergraph-of-entity indexes [§9.1.1], a study of rank stability for the nondeterministic random walk score [§9.1.2], a performance assessment for six hypergraph-of-entity models combining different index features [9.1.3], and a comparison with the graph-of-entity, commenting on the scalability of the models in regard to number of edges over the number of nodes [§9.1.4].
- **Section 9.2** describes the experiments carried during the TREC Common Core track, over the TREC Washington Post Corpus, where we measured the effectiveness of the hypergraph-of-entity. We compare two runs, one for a text-only model and another one for DBpedia based model that also contains entities and their relations [§9.2.1]. We then analyze the effectiveness for each run, as well as the best and worst queries, manually investigating some of the potential paths taken by random walks from the terms in those queries [§9.2.2].
- **Section 9.3** describes the assessment carried over the complete INEX 2009 Wikipedia collection, relying on the base mode of the hypergraph-of-entity, while exploring the random walk score as a universal ranking function, evaluating it for the three following tasks: ad hoc document retrieval, ad hoc entity retrieval, and entity list completion. We also argue that, for our model, the latter task is able to generalize related entity finding [§9.3.1]. We present a short overview on keyword extraction, identifying graph-based approaches [§9.3.2]. We describe the TextRank simplification used to build document profiles of a shorter length, and we also provide a study of the influence of the cutoff ratio applied to the selection of the top keywords [§9.3.3]. Finally, we describe the experimental framework, assessing the performance of each task over the joint representation model and the universal ranking function, and comparing it with the results for the most recent occurrences of the respective INEX tracks [§9.3.4].
- **Section 9.4** offers a reflection on retrieval heuristics, covering its pillar concepts [§9.4.1], using BM25 to exemplify how different heuristics influence those core concepts for supporting the creation of different ranking functions [§9.4.2], and commenting on the presence or absence of the pillars of information retrieval in the hypergraph-of-entity [§9.4.3].

## 9.1 JOINT REPRESENTATION MODEL EVALUATION: INEX 2009 WIKIPEDIA SUBSETS

We experimented with multiple variations of the hypergraph-of-entity, resulting in six different models: (i) the base model; (ii) the base model extended with *synonym* undirected hyperedges; (iii) the base model extended with undirected *context* hyperedges based on word embedding similarities; (iv) the base model extended with synonyms and then context; (v) the base model extended with context and then synonyms; and (vi) the base model extended with synonyms, context, and node and hyperedge weights[1]. Table 9.1 provides an overview of the tested models, showing

1 At this stage of the research, TF-bins had not been introduced yet, thus we could not include them here.

**Table 9.1:** Hypergraph-of-entity model overview. Superscript numbers beside the check marks indicate the integration order of a node or hyperedge in the model.

**(a)** Model nodes. Term nodes can be created based on the document, as well as expanded with synonyms external to the collection and contextually similar terms based on any corpus.

| **Model** | **term** | | | **entity** |
|---|---|---|---|---|
| | *Doc.* | *Syn.* | *Cont.* | |
| *Base Model* | ✓ | X | X | ✓ |
| *Syns* | ✓ | ✓ | X | ✓ |
| *Context* | ✓ | X | ✓ | ✓ |
| *Syns+Context* | ✓ | ✓(1) | ✓(2) | ✓ |
| *Context+Syns* | ✓ | ✓(2) | ✓(1) | ✓ |
| *Syns+Cont.+Weights* | ✓ | ✓(1) | ✓(2) | ✓ |

**(b)** Model hyperedges. Each hyperedge is depicted with either a tuple of sets (directed) or a single set (undirected). Elements that can be repeated are displayed with a subscript and elements that only appear once have no subscript. We use $t$ to represent term nodes and $e$ to represent entity nodes.

| **Model** | **document** $\{t_n, e_m\}$ | **contained_in** $(\{t_n\},\{e\})$ | **related_to** $\{e_m\}$ | **synonym** $\{t_n\}$ | **context** $\{t_n\}$ | **weight** |
|---|---|---|---|---|---|---|
| *Base Model* | ✓ | ✓ | ✓ | X | X | X |
| *Syns* | ✓ | ✓ | ✓ | ✓ | X | X |
| *Context* | ✓ | ✓ | ✓ | X | ✓ | X |
| *Syns+Context* | ✓ | ✓ | ✓ | ✓(1) | ✓(2) | X |
| *Context+Syns* | ✓ | ✓ | ✓ | ✓(2) | ✓(1) | X |
| *Syns+Cont.+Weights* | ✓ | ✓ | ✓ | ✓(1) | ✓(2) | ✓ |

which nodes and hyperedges were enabled for each model. In particular, it is relevant to notice that the integration order of synonyms and context matters — if we introduce synonyms and only then context, the term vocabulary might increase and word embeddings will also be computed for the synonym terms; on the other hand, if we introduce context and only then synonyms, the opposite might happen, given the word embeddings model has been trained with an external collection whose term vocabulary does not coincide with that from the original collection (this is not the case in the experiments we present here).

In the remainder of this section, we characterize an instance of the hypergraph-of-entity, with all the extensions, including synonyms, context and weights (Section 9.1.1), we study rank stability, since random walk score is not deterministic (Section 9.1.2) and, finally, we assess the performance of the hypergraph-of-entity representation and retrieval model, measuring effectiveness, as well as indexing and querying efficiency (Section 9.1.3).

### 9.1.1 Hypergraph-of-entity statistics

We characterize the hypergraph-of-entity representation for the INEX 2009 Wikipedia subset, indexed using the base model, with undirected *document* hyperedges, along with *synonym*, *context* and *weight* extensions. We begin by providing overall statistics regarding the number of nodes and hyperedges in the graph. We then analyze the connectivity power of *synonym* and *context* hyperedges, that is, their ability to establish new paths between documents. We finish by providing an overview of the weight distributions for different types of nodes and hyperedges.

Regarding disk space, the base model (the smallest index) required a total of 654 MiB for a collection of 203 MiB (compressed). Out of the 654 MiB, 540 MiB were used to store the hypergraph, 100 MiB to store node metadata and 15 MiB to store hyperedge metadata. On the other hand, the base model extended with synonyms, context and weights (the largest index) required a total of 715 MiB of space, out of which 582 MiB were used to store the hypergraph, 102 MiB to store node metadata,

**Table 9.2:** Number of nodes and hyperedges of the largest index (Syns+Cont.+Weights).

**(a)** Nodes.

| Node | Count |
|---|---|
| *term* | 1,126,685 |
| *entity* | 905,163 |
| **Total** | 2,031,848 |

**(b)** Hyperedges.

| Hyperedge | Count |
|---|---|
| *contained_in* | 784,672 |
| Total directed | 784,672 |
| *document* | 37,775 |
| *related_to* | 37,608 |
| *synonym* | 13,749 |
| *context* | 268,505 |
| Total undirected | 357,637 |
| **Total** | 1,142,309 |

19 MiB to store hyperedge metadata, 8 MiB to store node weights and 4.5 MiB to store hyperedge weights.

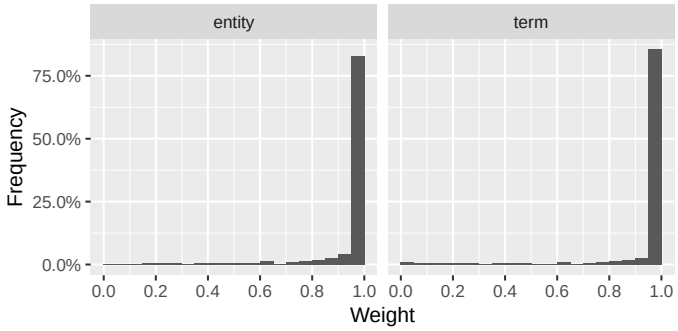
Table 9.2 shows the number of nodes and hyperedges for each type, also discriminating against direction. As we can see, the total number of hyperedges is significantly lower (almost half) than the number of nodes. This is the opposite behavior that we had found in the graph-of-entity, which didn't even include synonyms or contextual information. Most of the nodes in the hypergraph are used to represent terms, closely followed by entities. Most of the hyperedges are directed, specifically used to link terms and entities. Out of the undirected hyperedges, most are used to establish context — we might consider increasing the acceptance threshold for contextually similar terms, when building the word2vec similarity network, in order to lower the number of *context* hyperedges.

Relations of synonymy and contextual similarity were responsible for establishing new connections between documents, which in turn had the potential to improve recall over the base model. We analyzed the base model with synonyms and we found that synonyms established 6,968 new paths between documents, with 219.90 documents linked on average per synonym, with each synonym ranging between 1 and 12,839 linked documents. We did a similar analysis for the base model with context and we found that contextual similarity established 125,333 new paths between documents, with 53.71 documents linked on average through contextual similarity, ranging from 1 to 29,333 linked documents. The significantly higher number of new paths introduced by context, when compared to synonyms, might be explained by the fact that, despite only considering noun synonyms, potentially every word was a candidate for context extraction. On the other hand, we notice that, on average, context established a smaller number of links between documents than synonyms, despite the higher number of paths between each linked document.

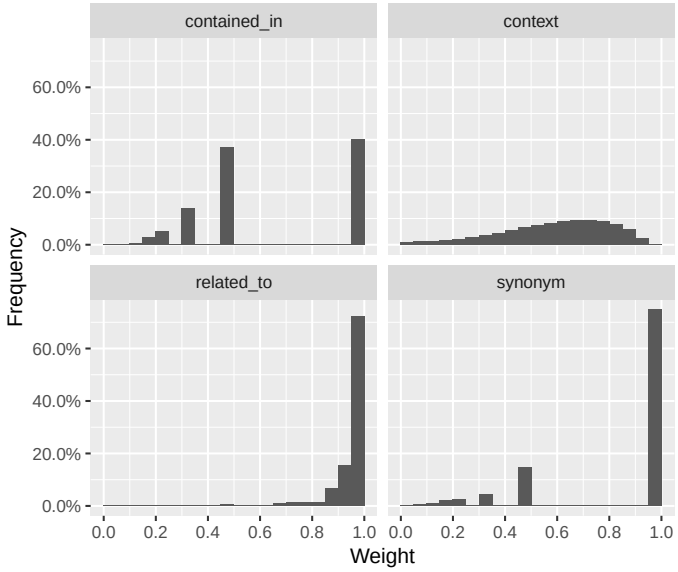
Figure 9.1 illustrates the distribution of node and hyperedge weights. As we can see, the selected weights are generally left skewed, showing a long left tail with most of the values within a range of 0.95 and 1.00 (note that we used a bin width of 0.05). This is less evident for *contained_in* hyperedges and does not happen for *document* hyperedges (not shown in the figure), since their weight is constant (0.5). Both *contained_in* and *synonym* hyperedge weight distributions have multiple missing ranges of values. This means that, granularity-wise these weighting functions are not ideal, regarding their discriminative power. In fact, this is also true for the remaining weighting functions.

### 9.1.2 Studying rank stability

While methods based on random walks usually converge to a limiting distribution, there is still a nondeterministic nature to this retrieval approach. This means that the probability distribution of visiting a set of nodes, given a departing set of seed




**(a)** Nodes.




**(b)** Hyperedges.

**Figure 9.1:** Hypergraph-of-entity weight distributions for INEX 2009 Wikipedia subset.

nodes, where random walkers start from, will eventually reach similar values for repeated experiments, given a sufficiently large number of iterations $r$. Measuring the performance of the random walk score only makes sense when in context with a rank stability analysis, through the measurement of rank convergence, for different runs with the same topic and parameter configuration.

We measured rank stability based on the Kendall's coefficient of concordance (Kendall's W) using fixed configurations of the random walk score as the ranking function. We repeated the same query multiple times, for a given configuration of $\ell$ and $r$, and then normalized each ranking list to ensure that they all contained the same set of documents. Missing documents were added to the end of the list, sorted by *doc_id* to ensure consistency in the calculation of Kendall's W, for equivalent rankings, with the same set of missing documents.

Table 9.3a summarizes the results for $\ell \in \{2, 3, 4\}$ and $r \in \{10, 50, 100\}$, using the geometric mean[1] over 100 repeats for each of the 52 topics of the INEX 2009 Wikipedia subset. For such low values of $r$, we did not find a significant difference in concordance, beyond a slight indication that a higher walk length $\ell$ tends to lower the concordance $W$. This is expected, since the longer the length of the walk, the higher the number of available path choices. More importantly, we found that, even for low values of $r$, we already achieve a concordance of over 80%. Nonaggregated

1 We used the geometric mean, since it is less sensitive to outliers and always smaller than the arithmetic mean, thus providing a more conservative result. However, for this particular case, the difference between arithmetic and geometric means was negligible.

**Table 9.3:** Measuring the stability of random walk score using Kendall's coefficient of concordance ($W$), for different parameter configurations.

**(a)** INEX 2009 Wikipedia subset (52 topics; 37,788 documents).

| $\ell$ | r | $W$ |
|---|---|---|
| 2 | 10 | 0.8719 |
| 2 | 50 | 0.8465 |
| 2 | 100 | 0.8450 |
| 3 | 10 | 0.8572 |
| 3 | 50 | 0.8312 |
| 3 | 100 | 0.8327 |
| 4 | 10 | 0.8439 |
| 4 | 50 | 0.8196 |
| 4 | 100 | 0.8224 |

**(b)** INEX 2009 Wikipedia smaller subset (3 topics; 2,234 documents).

| $\ell$ | r | $W$ | $W'$ |
|---|---|---|---|
| 2 | 100 | 0.7670 | 0.8386 |
| 2 | 1000 | 0.7646 | 0.9428 |
| 2 | 10000 | 0.9020 | 0.9857 |
| 3 | 100 | 0.7356 | 0.8733 |
| 3 | 1000 | 0.7881 | 0.9617 |
| 3 | 10000 | 0.9124 | 0.9901 |
| 4 | 100 | 0.7144 | 0.8957 |
| 4 | 1000 | 0.8178 | 0.9698 |
| 4 | 10000 | 0.9203 | 0.9930 |

values for Kendall's W, for each topic and parameter configuration, ranged from 0.7547 to 0.9521, with the first quartile already reaching 0.8030. Standard deviations were under 0.0521, showing stability over different topics. In order to better understand the behavior of concordance for higher values of r, we also replicated the experiment for the smaller subset with $r \in \{100, 1000, 10000\}$. Results, shown in Table 9.3b, illustrate the overall effect of increasing r — higher values of r result in a higher concordance. Even for low values of r, the results given by the random walk score are already considerably stable, which increases trust that a performance assessment should remain fairly unchanged for different runs with the same parameter configuration. Both tables show the concordance coefficient for $r = 100$, which is lower for the smaller subset. Given the geometric mean was calculated over only three topics, the influence of a single topic was quite impactful. In particular, we found that topic 2010023 (`[ retirement age ]`) resulted in a much lower concordance coefficient, ranging from 0.4544 to 0.7905. Further analysis of the remaining two topics showed that their concordance coefficients were in fact higher than the geometric mean depicts, ranging from 0.8189 to 0.9936. These values were also more in agreement with the experiment for the larger subset, as we can see from the geometric mean $W'$, calculated after removing topic 2010023. Based on the limited but consistent evidence of this analysis, where an incremental behavior of concordance was found for increasing values of r, we chose $r = 10^3$ as a good compromise that should provide an evaluation reliability of approximately 95%.

### 9.1.3 Assessing model performance

In the previous section, we have measured the stability of a ranking approach based on random walks. In this section, we describe how similar parameter configurations affect the performance of the retrieval model. In order to evaluate retrieval over the hypergraph-of-entity, we used the title of each topic from the INEX 2010 Ad Hoc Track as a search query. We then assessed effectiveness based on whether or not retrieved documents contained relevant passages, according to the provided relevance judgments. In order to measure efficiency, we also collected indexing and search times, as to understand the cost of using such a hypergraph-based representation, as well as different parameter configurations for the random walk score.

We tested each of the variations presented in Table 9.1, assessing the effectiveness of the Random Walks Score, using a combination of parameter configurations based on low walk lengths and high walk repeats, according to the intuition that closer nodes to the seeds (and therefore to the query) lead to more relevant documents/entities and that a higher number of repeats leads to convergence and therefore trustworthy results. We obtained the best hypergraph-of-entity MAP for the

**Table 9.4:** Best overall parameter configuration according to the mean average precision.

**(a)** Effectiveness (highest values for Lucene and hypergraph-of-entity in bold; differences in MAP are not statistically significant, except between the Lucene baselines and the hypergraph-of-entity indexes).

| Index | Ranking | GMAP | MAP | Precision | Recall | NDCG@10 | P@10 |
|---|---|---|---|---|---|---|---|
| Lucene | TF-IDF | 0.1345 | 0.1689 | **0.0650** | 0.8476 | 0.2291 | 0.2346 |
| | BM25 | **0.2740** | **0.3269** | 0.0647 | **0.8598** | **0.5607** | **0.5250** |
| Hypergraph-of-Entity: Random Walk Score ($\ell = 2$, $r = 10^3$) | | | | | | | |
| *Base Model* | RWS | 0.0285 | 0.0864 | 0.0219 | 0.8003 | **0.1413** | **0.1269** |
| *Syns* | RWS | 0.0281 | 0.0840 | 0.0225 | 0.8099 | 0.1301 | 0.1231 |
| *Context* | RWS | 0.0134 | 0.0811 | 0.0220 | 0.8027 | 0.1218 | 0.1192 |
| *Syns+Context* | RWS | 0.0299 | 0.0837 | 0.0236 | 0.8069 | 0.1310 | 0.1231 |
| *Context+Syns* | RWS | 0.0296 | 0.0814 | 0.0242 | **0.8148** | 0.1256 | 0.1250 |
| *Syns+Cont.+Weights* | RWS | **0.0313** | **0.0884** | **0.0274** | 0.8059 | 0.1256 | 0.1154 |

**(b)** Efficiency (lowest times for Lucene and hypergraph-of-entity in bold).

| Index | Ranking | Indexing Time | | Search Time | |
|---|---|---|---|---|---|
| | | Avg./Doc | Total | Avg./Query | Total |
| Lucene | TF-IDF | 2.16ms | 1m 21s 382ms | **1s 148ms** | **59s 698ms** |
| | BM25 | | | 1s 220ms | 1m 03s 461ms |
| Hypergraph-of-Entity: Random Walk Score ($\ell = 2$, $r = 10^3$) | | | | | |
| *Base Model* | RWS | 6.52ms | 4m 05s 612ms | **3m 22s 826ms** | **2h 55m 47s** |
| *Syns* | RWS | **6.22ms** | **3m 54s 587ms** | 3m 31s 038ms | 3h 02m 54s |
| *Context* | RWS | 6.35ms | 3m 59s 446ms | 3m 35s 623ms | 3h 06m 52s |
| *Syns+Context* | RWS | 6.29ms | 3m 57s 264ms | 3m 33s 000ms | 3h 04m 36s |
| *Context+Syns* | RWS | 6.33ms | 3m 58s 659ms | 3m 36s 487ms | 3h 07m 37s |
| *Syns+Cont.+Weights* | RWS | 6.52ms | 4m 05s 984ms | 10m 55s 590ms | 9h 28m 11s |

base model extended with synonyms, contextually similar terms and weights, with $\ell = 2$ and $r = 10^3$ (cf. Table 9.4a) — we verified that increasing values of $r$ suggested an increasing and plateauing performance. None of the hypergraph-of-entity variations were able to surpass the Lucene baselines, reaching MAP values between 0.0811 and 0.0884, when compared to 0.1689 for TF-IDF and 0.3269 for BM25. The best hypergraph-of-entity model according to GMAP, MAP and precision was *Syns + Cont. + Weights*, however the *Base Model* without extensions was able to reach the best results for NDCG@10 and P@10. We carried a Student's t-test for the 28 pairs of models, comparing average precisions for MAP and individual P@10 values per topic, using a p-value of 0.05. Results showed that the difference in MAP, as well as P@10, was statistically significant for TF-IDF and BM25, as well as for any Lucene baseline and any hypergraph-of-entity model, but not among different versions of our model.

The introduction of weights shows the flexibility of the model, in the sense that it is able to easily support the boosting of terms and entities, as well as the boosting of documents and other relations, in order to assign, for instance, a degree of certainty to each piece of information. Experiments also showed that the higher the walk length $\ell$, the worse the retrieval effectiveness. This is, by design, the expected behavior, since the further apart nodes are from the seed nodes (which represent the query), the less related to the query they are and thus the less relevant they are. The best recall for the hypergraph-of-entity was obtained for *Context + Syns* (0.8148), which was close to the baselines (0.8476 for TF-IDF and 0.8598 for BM25). The Geometric Mean Average Precision (GMAP) was included in Table 9.4a because it is less affected by outliers than MAP, thus providing additional insight. Through the comparison of GMAP and MAP, it becomes evident that a small number of topics are driving MAP up for the hypergraph-of-entity, despite many individual topics resulting in a low average precision — in some cases achieving values as low as zero (e.g., for topic 2010006 on the best *Context* model).

In Table 9.4b, we find the indexing and search times for the runs with the best MAP per variation. The hypergraph-of-entity took 2.9 times longer to index than Lucene, when comparing *Syns* with *Lucene*, as well as between 18.8 times longer to

**Table 9.5:** Graph-of-entity (GoE) vs hypergraph-of-entity (HGoE) with $\ell = 2$.

**(a)** Effectiveness (highest values for Lucene and graph-based models in bold).

| Index | Ranking | GMAP | MAP | Precision | Recall | NDCG@10 | P@10 |
|---|---|---|---|---|---|---|---|
| Lucene | TF-IDF | 0.1540 | 0.1710 | 0.1389 | 0.8007 | 0.2671 | 0.2800 |
| | BM25 | **0.2802** | **0.2963** | **0.1396** | **0.8241** | **0.5549** | **0.5000** |
| GoE | EW | 0.0003 | 0.0399 | **0.1771** | 0.2233 | 0.1480 | 0.1500 |
| HGoE | RWS($r = 10^1$) | 0.0000 | 0.0485 | 0.0734 | 0.3085 | 0.1229 | 0.1200 |
| | RWS($r = 10^2$) | 0.0546 | 0.1118 | 0.0342 | 0.7554 | 0.1474 | 0.1500 |
| | RWS($r = 10^3$) | 0.1017 | 0.1492 | 0.0199 | 0.9122 | **0.2074** | **0.2200** |
| | RWS($r = 10^4$) | **0.1224** | **0.1689** | 0.0167 | **0.9922** | 0.1699 | 0.1700 |

**(b)** Efficiency (lowest times for Lucene and graph-based models in bold).

| Index | Ranking | Indexing Time (Total) | Search Time (Avg./Query) | Nodes | Edges |
|---|---|---|---|---|---|
| Lucene | TF-IDF | **27s 769ms** | **209ms** | N/A | N/A |
| | BM25 | | 316ms | | |
| GoE | EW | 1h 38m | 21s 557ms | 981,647 | 9,942,647 |
| HGoE | RWS($r = 10^1$) | **53s 922ms** | **943ms** | 607,213 | 253,154 |
| | RWS($r = 10^2$) | | 11s 134ms | | |
| | RWS($r = 10^3$) | | 1m 17s 540ms | | |
| | RWS($r = 10^4$) | | 13m 04s 057ms | | |

query (best case scenario, for the *Syns* model with $\ell = 2$ and $r = 10^2$ and Lucene TF-IDF) and 1127 times longer to query (worst case scenario for *Syns + Cont. + Weights* with $\ell = 4$ and $r = 10^3$ and Lucene BM25 with $k_1 = 1.2$ and $b = 0.75$). Given the notable difference in efficiency between the weighed and non-weighted versions, it might be a good compromise to use the *Base Model* with $\ell = 2$ and $r = 10^3$, which is the most effective model when considering the top 10. Overall, search time was shown to range roughly between 9 and 23 minutes for $l = 4$ and $r = 10^3$ runs, with MAP scores between 0.06 and 0.08 and a coefficient of concordance around 0.82. However, if we consider $l = 2$ and a lower value $r = 10^2$, search time will drop to a range roughly between 22 seconds and 1 minute, with MAP scores of roughly 0.06 and a coefficient of concordance dropping to around 0.77. This means that we can achieve comparable effectiveness, while significantly increasing efficiency, despite compromising the concordance of multiple similar runs with the same parameter configuration (i.e., the ranking function won't converge).

### 9.1.4 Comparing graph-of-entity and hypergraph-of-entity

We compare the graph-of-entity with the hypergraph-of-entity, regarding effectiveness and efficiency, but also illustrate the difference in number of nodes and edges, particularly regarding the node-edge ratio. In order to better understand the differences in performance between the graph-of-entity and the hypergraph-of-entity, we were required to further reduce the size of the test collection. In particular, we used the INEX 2009 10T-NL Wikipedia subset, so that we were able to generate the graph-of-entity in a timely manner. As detailed in Section 4.1.1, sampling was based on the selection of 10 topics uniformly at random, filtering out documents that were not mentioned in the relevance judgments, and obtaining a collection of 7,487 documents (80% smaller than the subset based on 52 topics).

Table 9.5 compares the effectiveness and efficiency of graph-of-entity and hypergraph-of-entity, using Lucene as a baseline. For the graph-of-entity, we used the Entity Weight (EW) as the ranking function. For the hypergraph-of-entity, we used the base model (i.e., without synonyms, context or weights) and the Random Walk Score (RWS) as the ranking function. As we can see in Table 9.5a, hypergraph-of-entity is overall more effective than graph-of-entity, except when considering the macro averaged precision (Precision). As shown in Table 9.5b, hypergraph-of-entity

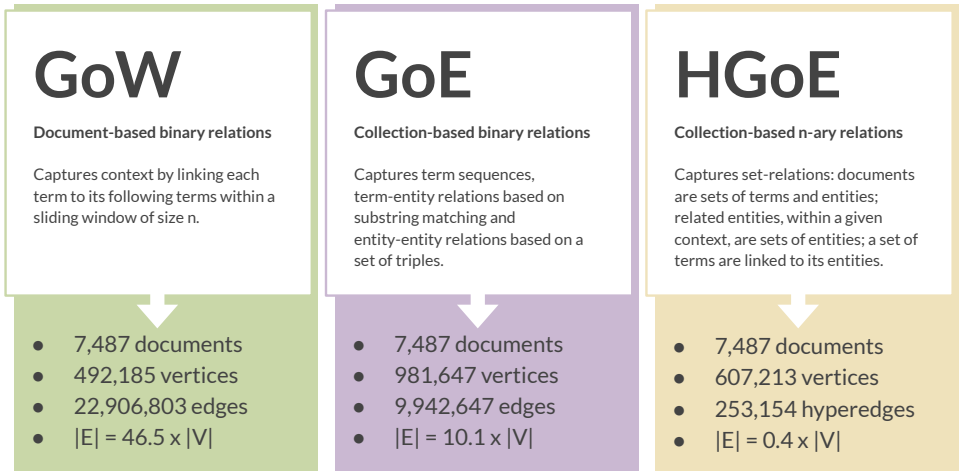



**Figure 9.2:** Comparing edge-node relation for graph-of-word, graph-of-entity, and hypergraph-of-entity (base model), over the INEX 2009 10T-NL Wikipedia subset.

is also considerably more efficient than graph-of-entity, taking only 53s 992ms to index when compared to 1h 38m for graph-of-entity. When analyzing search time for the hypergraph-of-entity, we can see that there is a tradeoff between effectiveness and efficiency that can be controlled through parameter $r$. For higher values, like $r = 10^4$, we reach a MAP score of 0.1689 but search time is a lot higher than the graph-of-entity (13m 4s when compared to 21s 557ms). On the other hand, for lower values, like $r = 10^1$ or even $r = 10^2$, where we reach MAP scores of 0.0485 and 0.1118 respectively, search time is lower than the graph-of-entity (943ms and 11s 134ms when compared to 21s 557ms). Additionally, by lowering the value of $r$, we also lower the rank stability, but even for $r \in \{10, 50, 100\}$ we were able to achieve coefficients of concordance of around 0.85 (cf. Table 9.3a), which might be an acceptable compromise efficiency-wise. The gain in indexing speed is particularly influenced by the growth in number of (hyper)edges when compared to the number of nodes. While the graph-of-entity has 10 times more edges than nodes, the hypergraph-of-entity has 2.4 times less edges than nodes. We also carried a Student's t-test for the 21 pairs of models, comparing average precisions for MAP and individual P@10 values per topic, using a p-value of 0.05. Results showed that the difference in MAP was statistically significant for TF-IDF and BM25, for BM25 and any hypergraph-of-entity model (except for $r = 10$), and for graph-of-entity and the Lucene baselines. When considering P@10, behavior was similar, except for TF-ID and any hypergraph-of-entity model, where the difference in P@10 was not statistically significant.

Figure 9.2 provides an overview of the three main graph-based models tested throughout this thesis. This includes GoW (Graph-of-Word), by Rousseau and Vazirgiannis [16], as well as GoE (Graph-of-Entity) and HGoE, proposed in this thesis. The graph-of-word was originally a document-based graph that, when converted into a collection-based graph, would reach nearly 500 thousand term nodes and over 22.9 million edges linking each term to the following three terms. This meant that, for only 7,487 documents, 46.5 edges were created for each of the 492,185 terms in the vocabulary. The original document-based approach proposed by Rousseau and Vazirgiannis resulted in a higher efficiency, since metrics could be computed per document, from each graph, and then stored in an inverted index. However, this model did not support the cross-referencing of information among different documents. The representation approach that we propose, as translated from the original model, had the sole purpose of establishing a baseline and a foundation for a collection-based graph. We further explored this foundation with the graph-of-entity, described in Chapter 6, where we only considered a single following term,

while also integrating entities. This resulted in an increased number of nodes, reaching nearly 1 million, however the number of edges was significantly reduced to nearly 10 million — 10.1 edges were created per node. Finally, in order to lower the complexity of graph traversals in the graph-of-entity, so that we could explore index extensions like synonymy or contextual similarity, we proposed the hypergraph-of-entity. This model was not only stored in memory instead of relying on a graph database, but it also significantly reduced the number of hyperedges for the number of nodes. This was possible by simplifying relations and promoting the grouping of multiple nodes. As a result, for 607,213 nodes, only 253,154 hyperedges were created, which means that less than one hyperedge was created per node — two hyperedges were created for every five nodes. Since the index size of a collection of documents is bound by the size of its vocabulary — at the very least the unique set of all words, or a selection of keywords, must be a part of the index — being able to control how (hyper)edges scale is a essencial to control performance and reduce complexity. It is a matter of how much information can we cross-reference within a given representation model, to better solve an information need, without losing the complexities provided by captured relations.

## 9.2 TEXT-ONLY VS JOINT REPRESENTATION MODEL EVALUATION: TREC WASHINGTON POST CORPUS

We describe our participation in the TREC 2018 Common Core track, where we experimented with hyperedge-based document ranking, over the hypergraph-of-entity. In this work, we use the hypergraph-of-entity as an indexing data structure for the TREC Washington Post Corpus, where we represent terms, entities and their relations. For the base models, we explore both a text-only representation (*feup-run1*) and a joint representation of text and knowledge, based on entities and relations from DBpedia (*feup-run2*). We also collaborated with the University of Alicante to experiment with reranking for diversity, based on the maximal marginal relevance and document profiling. This second part of the work is described in the notebook paper [270]. We evaluated retrieval effectiveness over the hypergraph-of-entity based on the mean average precision, over the relevance judgments provided by TREC. Our best results were for *feup-run1*, with a MAP score of 0.0070 and a P@10 of 0.0680.

We assessed retrieval effectiveness based on the relevance judgments provided by TREC and based on the 50 test topics for 2018. The provided test topics consisted of 25 topics from the 2017 Common Core track, as well as 25 new topics prepared by NIST assessors. All experiments were run over the TREC Washington Post Corpus using the title of the topic as a keyword query. Each submitted run was assigned a priority, and participants were assured that at least two runs per team would be judged, at the very least considering the top 10 documents per topic. We assigned the top priority to the base runs (*feup-run1* and *feup-run2*), simply because the remaining runs resulted from a reranking of the same set of documents, which had a high probability of sharing a similar set of top 10 documents, given MMR (Maximal Marginal Relevance) was used for reranking. In Section 9.2.1, we describe the submitted runs, as well as statistics about the hypergraph-of-entity indexes used. In Section 9.2.2, we analyze the evaluation results from TREC 2018 Common Core track regarding the two relevant runs for this thesis.

### 9.2.1 Submitted runs

We prepared and submitted two runs to TREC Common Core track. The first run (*feup-run1*) was based on a text-only version of the hypergraph-of-entity, simply consisting of *term* nodes and *document* hyperedges. The second run (*feup-run2*) was

**Table 9.6:** Statistics for the hypergraphs-of-entity used in *feup-run1* and *feup-run2*.

| Version | Statistic | Value |
|---|---|---|
| Text-only | *term* nodes | 886,298 |
| | **Total nodes** | **886,298** |
| | undirected *document* hyperedges | 595,037 |
| | **Total hyperedges** | **595,037** |
| DBpedia | *term* nodes | 886,298 |
| | *entity* nodes | 276,735 |
| | **Total nodes** | **1,163,033** |
| | undirected *document* hyperedges | 595,037 |
| | undirected *related_to* hyperedges | 595,037 |
| | Total undirected hyperedges | 1,190,074 |
| | directed *contained_in* hyperedges | 266,962 |
| | Total directed hyperedges | 266,962 |
| | **Total hyperedges** | **1,457,036** |

an extension of the text-only version, where we added DBpedia [50] entities and the respective triples for each entity, relying on *entity* nodes, *related_to* hyperedges and *contained_in* hyperedges for the representation. The information extraction process of *feup-run2* was limited to the first three paragraphs of each document, due to resource constraints — our implementation requires the hypergraph-of-entity to be fully loaded into RAM and we were, at this point, unable to consider all triples for all extracted entities using the available 32 GiB of RAM. Named Entity Recognition (NER) was carried based on the Aho-Corasick string-searching algorithm [352] over a combined list of all *'@en' rdfs:label* for *#dbo:Person*, *#dbo:Organisation* and *#dbo:Place* entities. An HTTP endpoint implementing this strategy is available in Army ANT[1]. It can be run by first setting up `defaults/service/ner/entity_list` in *config.yaml* and launching server, and then sending a `POST` request to `http://localhost:8080/service/ner`, using the field *text* to get the list of identified entities. This NER strategy was chosen so that we could more efficiently match a finite list of labels with each document.

Table 9.6 shows several statistics for the hypergraphs-of-entity used in runs 1 and 2. No stemming or lemmatization was applied to either version, resulting in a vocabulary of over 800 thousand terms. When extended with DBpedia information, over 200 thousand people, organizations and places were extracted as entities. As expected, there were as many *document* hyperedges as documents in the collection. We also included *related_to* hyperedges to model co-occurrence of entities in documents, but this should be improved in the future to better take advantage of triples associated with each document and its entities. Finally, over 200 thousand relations were established between sets of terms and their corresponding entity — this was based on term matching with the entity name, but it could easily use another type of term–entity association instead, or be extended to consider other languages, for cross-language retrieval.

### 9.2.2 Retrieval effectiveness

The command `trec_eval -c -q -M1000` was used by TREC to calculate multiple effectiveness metrics, where `-c` ensures that the average is done over the complete set of queries in the relevance judgments, `-q` includes per-topic evaluations, and `-M1000` considers only a maximum of 1,000 retrieved documents. Out of the provided metrics, we selected the Mean Average Precision (MAP) as the overall effectiveness indicator and also included the Geometric Mean Average Precision (GMAP), the Normalized Discounted Cumulative Gain at a cutoff of p (NDCG@10), and the Precision at a cutoff of n (P@10) as a complementary indicators.

1 https://github.com/feup-infolab/army-ant/tree/trec-2018

**Table 9.7:** Evaluation of TREC 2018 Common Core track runs.

| Run | GMAP | MAP | NDCG@10 | P@10 |
|---|---|---|---|---|
| *feup-run1* | 0.0001 | 0.0070 | 0.0572 | 0.0680 |
| *feup-run2* | 0.0002 | 0.0051 | 0.0227 | 0.0240 |

**Table 9.8:** Characteristics of best and worst retrieved topics.

| Run(s) | Topic | Best MAP | Query | Rel. Retr. | Rel. | Node Matches | |
|---|---|---|---|---|---|---|---|
| | | | | | | Text-Only | DBpedia |
| | | | **Best** | | | | |
| 1 | 810 | 0.1429 | `[ diabetes and toxic chemicals ]` | 1 | 7 | 15-NA-52-10 | 21-NA-62-15 |
| 2 | 822 | 0.1312 | `[ Sony cyberattack ]` | 34 | 43 | 24-6 | 78-6 |
| 1 | 823 | 0.0294 | `[ control of MRSA ]` | 6 | 58 | 93-NA-2 | 164-NA-2 |
| | | | **Worst** | | | | |
| 1 | 341 | 0.0000 | `[ Airport Security ]` | 0 | 124 | 32-67 | 621-238 |
| 1, 2 | 427 | 0.0000 | `[ UV damage, eyes ]` | 0 | 28 | NA-10-109 | NA-17-295 |
| 1, 2 | 819 | 0.0000 | `[ U.S. age demographics ]` | 0 | 15 | 7,949-3,481-2 | 12,850-5,896-2 |

As we can see in Table 9.7, the best run was *feup-run1*, with a MAP of 0.0070 and a P@10 of 0.0680. The difference between MAP scores for the two runs is not statistically significant. Additionally, it is clear that, overall, precision was quite low, with a MAP score achieving a maximum of 0.1429 for topic 810 in run 1, and the next best score achieving a MAP of 0.1312, for topic 822 in run 2. Following topics 810 and 822 there is topic 823 with a lower MAP score of 0.0294 for run 1. Over half of the topics for all runs resulted in a MAP score of zero. This includes for instance topic 824 for run 1, or topic 819 for run 2.

Table 9.8 shows the keyword queries corresponding to the best and worst topics according to MAP, over the two runs. The best results, for topic 810, were based on the text-only version of the hypergraph-of-entity. According to this small sample, there is no clear advantage of using the DBpedia version over the text-only version of the hypergraph-of-entity, or vice-versa. This might become clearer with a better usage of the triples associated with the documents, when available. The table also shows the number of relevant documents retrieved (*Rel. Retr.*) versus the number of relevant documents in the collection (*Rel.*), as well as the number of partial node matches in both versions of the hypergraph-of-entity, so that we better understand the amount of information potentially covering each term (i.e., how many nodes are there for each keyword in the query). As we can see, with the exception of topic 822 (`[ Sony cyberattack ]`) very few relevant documents were retrieved. Additionally, the highest MAP score is justified by a low overall number of relevant documents. When looking at the number of matching nodes per query term, we found that this is also not an indicator of effectiveness, as the worst topics can either have a high or low number of matching nodes.

A manual investigation, however, provided a few insights as to what might have caused such a low overall precision. First, we found a tokenization issue where stopwords weren't sometimes split from other words (e.g., "and.hacking", "economics.and"), needlessly extending the vocabulary and discarding paths for random walks. This was also the first time we indexed a collection of news articles, after having worked with encyclopedic content, and we found that random walks of length $\ell = 2$ would easily reach unrelated topics. For example, when departing from "diabetes", we would step into an article entitled "Can running help autistic children?" and then randomly into the term "megan", which would lead to "Home sales in Loudoun and Fauquier counties". The constraints provided by the hypergraph-of-entity are still not enough, in particular to support search using random walks over a collection of news articles. Several approaches might be taken to improve this, namely introducing *sentence*, *paragraph* or *passage* hyperedges in order to avoid taking steps into unrelated directions (such as "megan"). Obviously, de-

**Table 9.9:** Best runs per team for TREC 2018 Common Core track.

| Team | Run ID | Type | MAP |
|---|---|---|---|
| UWaterlooMDS | UWaterMDS_Rank | Manual | 0.4303 |
| RMIT | RMITUQVDBFNZDM1 | Manual | 0.3850 |
| h2oloo | h2oloo_enrm30.6 | Automatic | 0.3382 |
| MRG_UWaterloo | uwmrg | Automatic | 0.2761 |
| Anserini | anserini_qlax | Automatic | 0.2749 |
| Sabir | sab18coreE1 | Feedback | 0.2510 |
| NOVASearch | bt-BoWBoE | Feedback | 0.2468 |
| UMass | umass_sdm | Automatic | 0.2339 |
| JARIR | jarir_sg_re | Automatic | 0.2040 |
| Webis | webis-argument | Automatic | 0.1015 |
| FEUP | feup-run1 | Automatic | 0.0070 |

spite document scoring depending on $r = 1{,}000$ random walks for each seed node (frequently multiple entities for a single term), allowing such unrelated walks is still detrimental to the overall ranking. Furthermore, the hypergraph-of-entity does not support any type of document length normalization, which is also affecting the quality of random walks. We also did not use any stemming or lemmatization, since we wanted to leave room for the exploration of syntactic relations, which could only be extracted and modeled based on complete sentences. Finally, due to scalability issues, we only relied on the first three paragraphs of each document, which was rather detrimental to retrieval performance, reducing the chances for matching terms, thus leading to the anomalous low MAP. Additionally, this is also a more challenging test collection:

> *There are many fewer relevant documents in 2018 compared to 2017 [by design], and the percentage of uniquely retrieved relevant documents to relevant documents is also much smaller in 2018 than in 2017.*
>
> – Ellen Voorhees, TREC Core Google Group, October 4, 2018

Accordingly, the results from our participation in TREC 2018 Common Core track were largely inconclusive and difficult to compare with the runs from the remaining teams.

Table 9.9 shows the best runs, according to MAP, for each participant in TREC 2018 Common Core track. Runs are classified as *Manual* (a human selection of ranked documents), *Automatic* (a retrieval approach according to a predetermined ranking model), and *Feedback* (a retrieval approach that relies on existing relevance judgments). As we can see, the best result was a manual ranking provided by *UWaterlooMDS* that resulted in a MAP of 0.43. Results that relied on pre-existing feedback occupied a middle position regarding retrieval effectiveness, with the remaining being automatic runs. Our best run placed last due to the reasons that we listed above, namely the reduced number of indexed paragraphs in comparison with the approaches from the remaining participants [353–361].

## 9.3 UNIVERSAL RANKING FUNCTION EVALUATION: INEX 2009 WIKIPEDIA COLLECTION

In the previous two sections, we focused on measuring the effectiveness of the random walk score over a single task — ad hoc document retrieval — while measuring the impact of different configurations of the hypergraph-of-entity joint representation model. In this section, however, we focus on the other aspect of our general model for entity-oriented search, the universal ranking function. In particular, we test the random walk score over three different tasks, ensuring that it provides a

consistent performance with the baselines for each task. In the following sections, we delve deeper into this analysis. In Section 9.3.1, we describe the overall evaluation approach. In Section 9.3.2, we provide an overview on keyword extraction techniques, which we apply in Section 9.3.3 to reduce the vocabulary size, as well as the number of entities to represent using the hypergraph-of-entity — this enables us to index the complete INEX 2009 Wikipedia collection. Finally, in Section 9.3.4, we present and comment on the performance of our general retrieval model over three different entity-oriented search tasks. We rely on a single index and ranking function, merely controlling the input and output nodes and hyperedges to respond to different tasks.

### 9.3.1 Approaching the evaluation of a general retrieval model

The hypergraph-of-entity was proposed as a joint representation model for text, entities and their relations, with the random walk score as a universal ranking function. This retrieval model provides a starting point not only for exploring the computation of multiple tasks from entity-oriented search, but also for finding answers based on a common source of cross-referenceable information. In this section, we explore the potential of the hypergraph-of-entity for ad hoc document retrieval, ad hoc entity retrieval, and for entity list completion, reinforcing the generalization line that has been proposed.

Out of the four proposed tasks in Section 1.3.2, we do not directly experiment with related entity finding, for two reasons. First, there is no dataset combining a corpus with a knowledge base that provides topics and relevance judgments for the four tasks — we use the INEX 2009 Wikipedia collection, which provides relevance judgments for the ad hoc document retrieval, ad hoc entity retrieval, and entity list completion. Secondly, we argue that entity list completion is a generalization of related entity finding, where the latter only takes one entity as input, while the former takes one entity, as well as examples of related entities, that work as relevance feedback. Both related entity finding, and entity list completion are defined as follows. Given an entity, a target type, and a relation, find other entities of the given target type that respect the specified relation. The difference between the two tasks lies in the fact that entity list completion also includes example entities to drive results towards similar entities. We opted, however, to simplify the view on these tasks, defining entity list completion as the task of finding other similar entities, given a set of input entities. For the particular case when this set has unitary cardinality, we consider it to be the same as related entity finding. This simplified definition makes particular sense in the context of hypergraph-of-entity, as the model does not store entity or relation types, making any type restrictions useless.

In order to be able to index the complete INEX 2009 Wikipedia collection, we are also required to reduce the size of the hypergraph-of-entity. Since we do not control the number of documents in a collection, we retain only representative keywords for each document. This not only reduces the quantity of nodes — and, indirectly, the quantity of hyperedges — but it also improves the overall quality of the model and its retrieval effectiveness. Accordingly, the contribution of this work is two-fold.

- We tackle the performance issues of the hypergraph-of-entity by reducing the number of terms in each document through keyword extraction;
- We assess the performance of a universal ranking function for three entity-oriented search tasks:
  - Ad hoc document retrieval;
  - Ad hoc entity retrieval;
  - Entity list completion.

We rely on a simplified version of TextRank, for keyword extraction. This version is more efficient but less effective than the original, which still results in an adequate amount of text usable for indexing. Different tasks are not comparable among each other. They are only comparable with their corresponding baselines. Nevertheless, our main goal is to understand whether performance is consistent and acceptable. Thus, our experiments are based on a common dataset, indexed using the hypergraph-of-entity. A single index is used to evaluate three different tasks, based on a universal ranking function and a set of three separate relevance judgments.

### 9.3.2 An overview on keyword extraction

Keyword extraction is the process of identifying significant or descriptive words or short expressions to illustrate a given document. In the hypergraph-of-entity, both words and entities are represented as nodes, while relations are represented as hyperedges. The number of words that exist in a language is finite and the number of entities can also be considered finite, particularly for a given snapshot of a knowledge base, which is often used. This means that, as the index grows, the number of nodes will eventually converge. One way to reduce the size of the hypergraph-of-entity is to limit the number of nodes it contains. We experimented with keyword extraction to reduce document length and therefore the number of term nodes in the model. In this section, we explore several keyword extraction approaches, distinguishing between non-graph-based [8, 9, 362] and graph-based [111, 363–365].

Table 9.10 provides a chronological overview of keyword extraction approaches. As we can see, many approaches are based on a graph of terms, which are frequently filtered by part-of-speech (POS) tags, in particular retaining nouns and adjectives. In TextRank [363], they used and undirected graph; in SingleRank [364], they also used an undirected graph, but they also included terms from similar documents; in RAKE [365], they used an undirected weighted graph; and in graph-of-word [111], they used a directed graph. TextRank and SingleRank both relied on PageRank, while RAKE experimented with degree, and graph-of-word with maximal k-core retention. TextRank, SingleRank and RAKE also considered a post-processing stage, where keywords were merged into multi-term keywords. Out of the two non-graph-based approaches we covered, TF-IDF serves to illustrate one of the first, most iconic metrics of term importance, while YAKE! shows a state-of-the-art approach, based on other features that are not easily represented as a graph. In particular, these include: casing (ratio of uppercase term frequency to term frequency), word position (based on the median position of sentences containing the word), word frequency (divided by the sum of the mean and standard deviation), word relatedness to context (measuring the diversity of words co-occurring within a left and right window), and word DifSentence (the normalized number of sentences containing the word).

For the hypergraph-of-entity, the best keyword extraction method is not the most effective, but the most efficient. When working with general models, we must also consider whether the approach fits our current framework, as to prepare for a future integration leading to improved generality. Taking this into account means that, for our retrieval model, random walk and graph-based approaches are preferred. This leaves both TextRank and SingleRank as ideal candidates, since they both rely on graphs and PageRank, a random walk based approach. We selected TextRank, as SingleRank would expand to similar documents too prematurely for our model. This would have represented not only additional overhead, but also a redundant step that would have been analogously taken by the random walk score in hypergraph-of-entity during search. In Section 9.3.3, we present the details on how we further modified TextRank to reduce computation time, with very little impact in effectiveness for our retrieval model.

**Table 9.10:** Chronological overview of keyword extraction algorithms, identifying graph-based (GB) approaches.

| Algorithm | Year(s) | GB | Description |
|---|---|---|---|
| TF-IDF [8, 9] | 1957-72 | ✗ | Document keywords selected based on whether they are frequent in the document, but rare in the collection. |
| TextRank [363] | 2004 | ✓ | An undirected graph is built based on the co-occurrence of terms, optionally filtered by POS tags, within a sliding window. PageRank is applied, terms are ranked and adjacent terms form multi-term keywords. |
| SingleRank [364] | 2008 | ✓ | The input document is expanded with k other similar documents. An undirected graph is built to link syntactically filtered words above a given affinity (weighted average of term frequencies per document similarity) threshold. Saliency is then computed based on the PageRank of this graph and adjacent top terms are merged into multi-term keywords. |
| RAKE [365] | 2010 | ✓ | Candidate keywords are generated by splitting the document by stopwords. A weighted graph of keyword co-occurrence is used to compute a keyword score (term frequency, degree and tf-degree ratio were tested as weighting functions). Wrongly split keywords are merged based on whether that instance repeats in the document and their scores are summed. |
| Graph-of-Word [111] | 2015 | ✓ | A directed graph is built based on a sliding window, filtered by noun and adjective POS tags. Keywords are then selected from the main core (i.e., largest the k-core). A k-core is a subgraph where every node has degree at least k. This way, there is no need to define a cutoff based on a threshold or ratio. |
| YAKE! [362] | 2018 | ✗ | Several term weighting functions are proposed and combined to score candidate keywords: casing, word position, word frequency, word relatedness to context, and word DifSentence. Multi-word keywords are then considered through the combination of candidate keyword scores, eliminating similar candidates through the Levenshtein distance. |

### 9.3.3 Simplifying TextRank for efficiency

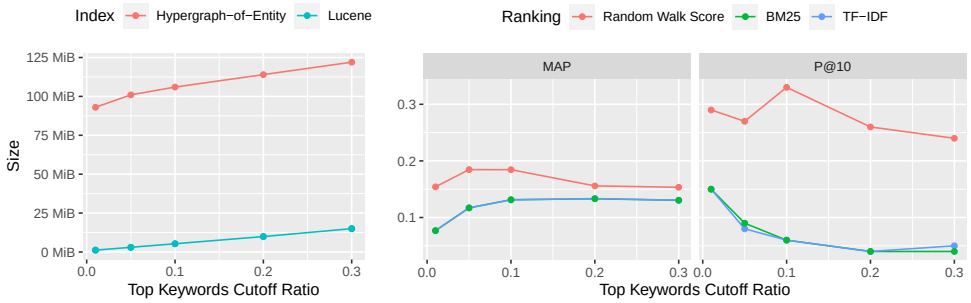
We use a simplified version of TextRank to build document profiles, based on the keywords extracted from each article in the INEX 2009 Wikipedia collection. This way, we obtain shorter documents that are representative of the original Wikipedia articles, but require less space during indexing. We also study the behavior of the *ratio* parameter, based on a smaller subset of the test collection, in order to determine the ideal fraction of keywords required for a good performance. Keyword extraction is done using a simplified version of TextRank [363] over the preprocessed text (i.e., lower case, tokenized, without stopwords). We ignore POS tagging, syntactic filtering, and keyword collapse. Each pair of terms within a sliding window of size $n = 4$ is represented as two nodes connected in an undirected graph. PageRank is then computed for this graph, and term nodes are ranked accordingly. A fraction of the top keywords, defined by a *ratio* parameter, is then used to represent the document in the index.

In order to select the ideal fraction of keywords to use, we experimented with different ratio values based on the INEX 2009 10T-NL Wikipedia subset, for the ad hoc document retrieval task. We compared the size of the generated index, in bytes as well as number of nodes and edges, for several ratio values: 0.01, 0.05, 0.10, 0.20, 0.30. For each run, we computed several performance metrics: P@10, NDCG@10, MAP, GMAP. As we can see in Table 9.11, keyword extraction results in a considerable reduction, particularly for Lucene, where the index is 6.9× smaller

**Table 9.11:** Evaluating performance for top keyword cutoff ratios ranging from 1% to 30%. In bold we show the best values per ranking function, over keyword based runs (i.e., ignoring full-text, as identified by missing cutoff ratios).

| Index | Ranking | Ratio | Size | Nodes | Edges | P@10 | NDCG@10 | MAP | GMAP |
|---|---|---|---|---|---|---|---|---|---|
| Lucene | TF-IDF | 0.01 | **1.2 MiB** | – | – | **0.1500** | **0.1674** | 0.0769 | 6.50e-06 |
| | | 0.05 | 3.0 MiB | – | – | 0.0800 | 0.0720 | 0.1172 | 1.00e-05 |
| | | 0.10 | 5.3 MiB | – | – | 0.0600 | 0.0477 | 0.1310 | 1.31e-05 |
| | | 0.20 | 9.9 MiB | – | – | 0.0400 | 0.0316 | **0.1335** | 1.31e-05 |
| | | 0.30 | 15 MiB | – | – | 0.0500 | 0.0380 | 0.1306 | **1.43e-05** |
| | | – | 104 MiB | – | – | 0.2800 | 0.2667 | 0.2160 | 1.76e-01 |
| | BM25 $k_1$=1.2,b=0.75 | 0.01 | **1.2 MiB** | – | – | **0.1500** | **0.1674** | 0.0768 | 6.48E-06 |
| | | 0.05 | 3.0 MiB | – | – | 0.0900 | 0.0734 | 0.1169 | 9.99E-06 |
| | | 0.10 | 5.3 MiB | – | – | 0.0600 | 0.0472 | 0.1315 | 1.31E-05 |
| | | 0.20 | 9.9 MiB | – | – | 0.0400 | 0.0285 | **0.1331** | 1.30E-05 |
| | | 0.30 | 15 MiB | – | – | 0.0400 | 0.0285 | 0.1304 | **1.43E-05** |
| | | – | 104 MiB | – | – | 0.4900 | 0.5479 | 0.3412 | 3.15E-01 |
| HGoE | RWS $\ell$=2,r=1000 exp.=false | 0.01 | **93 MiB** | **291k** | **146k** | 0.2900 | 0.2581 | 0.1542 | 9.22e-02 |
| | | 0.05 | 101 MiB | 301k | 177k | 0.2700 | 0.2724 | **0.1846** | **1.07e-01** |
| | | 0.10 | 106 MiB | 312k | 190k | **0.3300** | **0.3227** | 0.1845 | 1.06e-01 |
| | | 0.20 | 114 MiB | 334k | 204k | 0.2600 | 0.2299 | 0.1559 | 9.09e-02 |
| | | 0.30 | 122 MiB | 360k | 214k | 0.2400 | 0.2484 | 0.1534 | 9.36e-02 |
| | | – | 182 MiB | 607k | 253k | 0.1700 | 0.1671 | 0.1312 | 1.01e-01 |

for the top 30% keywords. Similarly, the hypergraph-of-entity is reduced in size, resulting in a 1.5× smaller index for the top 30% keywords. As we can see in Figure 9.3a, the size of the index decreases with the ratio, but always achieves a better reduction for Lucene. When considering the top 1% keywords, the Lucene index is reduced 86.7×, from 104 MiB to 1.2 MiB, while the hypergraph-of-entity is only reduced 2.0×, from 182 MiB to 93 MiB. In particular, the number of nodes is reduced 2.1×, from 607 to 291 thousand nodes, while the number of hyperedges is reduced 1.7×, from 253 to 146 thousand hyperedges. When considering the Mean Average Precision (MAP), the complete indexes (i.e., with all document terms) achieve the best performance for Lucene with BM25. However, when considering any ratio, as show in Figure 9.3b, the version reduced to a document profile based on the top keywords consistently achieves a better MAP for the random walk score (0.18 for the top 1% and top 5% keywords). While the overall performance is lower, the random walk score is able to outperform TF-IDF and BM25, when available information is limited and, perhaps, more representative or selective. In fact, the lower the ratio, the better the MAP, reaching the ideal reduction and performance with the top 5% keywords, after which MAP starts decreasing again, as there is too little information in the top 1% keywords to discriminate the documents. Based on the results of this small-scale experiment, we opted for a ratio of 0.05, despite 0.10




**(a)** Storage efficiency.

**(b)** Retrieval effectiveness.

**Figure 9.3:** Evolution of performance metrics for increasing cutoff ratios of top keywords.

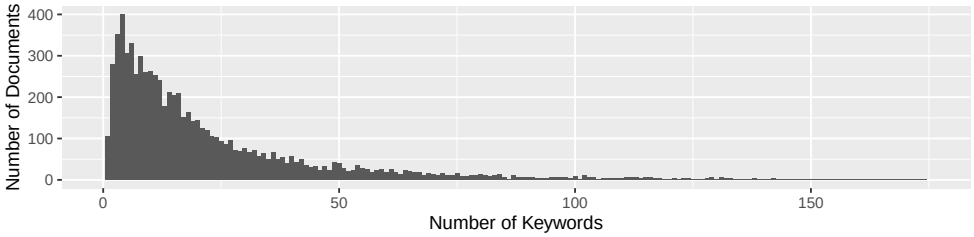



**Figure 9.4:** Keyword distribution for INEX 2009 10T-NL.

having reached both a higher NDCG@10 and P@10 (cf. Figure 9.3b). Our goal was to prioritize the minimization of space requirements.

Based on the simplified version of TextRank, the absolute number of keywords, retained based on the top 5% per document, were distributed as illustrated in Figure 9.4. On average, there were 22.9 keywords per document. There were 105 documents represented by only one keyword, 280 documents represented by two keywords, and 351 documents represented by three keywords. Most of the documents (400) were represented by four keywords, from then on following a logarithmic distribution, up to a maximum of 361 keywords (not displayed in the plot for readability).

Keyword-based reduction directly impacted the number of nodes and indirectly impacted the number of hyperedges, since it reduced the number of links between terms and entities, due to a lower number of terms being considered. In our experiments, we used the base model of the hypergraph-of-entity, without synonymy or contextual similarity relations. This resulted in a hypergraph with 3,506,823 nodes (633,269 terms and 2,873,554 entities), as well as 7,721,743 hyperedges (2,653,452 documents, 2,629,544 entity relations using subject-based grouping, and 2,438,747 text-entity relations based on the term occurrence in the entity name).

### 9.3.4 Assessing the random walk score as a universal ranking function

While the hypergraph-of-entity is conceptually able to support multiple tasks, their individual performance, over a common index, still needs to be tested. In order to better understand the viability of the hypergraph-of-entity as a general model for entity-oriented search, we assess the effectiveness and efficiency for the following three tasks: (1) ad hoc document retrieval (based on topics and qrels from INEX 2010 Ad Hoc track [248]); (2) ad hoc entity retrieval (based on topics and qrels from INEX 2009 Entity Ranking track [247], for the entity ranking task); and (3) entity list completion (based on topics and qrels from INEX 2009 Entity Ranking track, for the list completion task). The approaches we describe here, including baselines, were developed as a part of Army ANT [366]. This framework is available as open source software[1] and it can be used to reproduce these experiments[2]. The runs for each task were issued according to Table 7.4, using the parameter configuration from Table 7.5 — i.e., $\ell = 2$, $r = 10^4$, $\Delta_{nf} = 0$, $\Delta_{ef} = 0$, *exp.* $=$ F, *dir.* $=$ T, and *wei.* $=$ F — except for TF-bins, where *wei.* $=$ T was used, assigning a default weight of 0.5 to otherwise unweighted nodes and hyperedges.

Runs were based on the hypergraph-of-entity, built over the top 5% keywords, extracted from each document of the complete INEX 2009 Wikipedia collection (Section 4.1.1), through TextRank. We also considered different versions of the hypergraph-of-entity: the *Base Model*, without index extensions; the *Syns* model, which added *synonym* hyperedges, per term, based on WordNet synsets; the *Context* model, which added *context* hyperedges, per term, based on the most similar other

1 Army ANT is available at: https://github.com/feup-infolab/army-ant/tree/develop.
2 Please consult our YouTube videos to learn how to setup the framework: https://www.youtube.com/playlist?list=PLc6NtbG0dqo1wGoYdTZkVd7I4SFNodKot.

**Table 9.12:** Evaluation results for hypergraph-of-entity as a general retrieval model (best scores per task in bold).

| Index | Task | Ranking | Avg./query | MAP | GMAP | P@10 | NDCG@10 |
|---|---|---|---|---|---|---|---|
| | | | **Lucene** | | | | |
| Doc. Index | Ad hoc document retrieval | TF-IDF | 460ms | 0.0228 | 0.0000 | 0.0692 | 0.0778 |
| | | BM25 | 370ms | 0.0324 | 0.0000 | 0.1173 | 0.1274 |
| Ent. Index | Ad hoc entity retrieval | TF-IDF | 1s 370ms | 0.0373 | 0.0000 | 0.0636 | 0.0670 |
| | | BM25 | 798ms | 0.0668 | 0.0000 | 0.1182 | 0.1165 |
| | Entity list completion | TF-IDF | 1s 230ms | 0.0558 | 0.0044 | 0.1000 | 0.1014 |
| | | BM25 | 1s 221ms | 0.0666 | 0.0067 | **0.1250** | **0.1212** |
| | | | **Hypergraph-of-Entity** | | | | |
| Base Model | Ad hoc document retrieval | RWS | 23s 405ms | 0.0863 | 0.0278 | 0.2462 | 0.2662 |
| | Ad hoc entity retrieval | | 26s 330ms | **0.1390** | 0.0002 | 0.2455 | 0.2425 |
| | Entity list completion | | 19s 162ms | 0.0879 | **0.0376** | 0.0769 | 0.0594 |
| Syns | Ad hoc document retrieval | RWS | 55s 555ms | **0.0937** | **0.0303** | 0.2615 | 0.2812 |
| | Ad hoc entity retrieval | | 30s 232ms | 0.1337 | **0.0004** | 0.2473 | **0.2445** |
| | Entity list completion | | 19s 875ms | 0.0857 | 0.0368 | 0.0635 | 0.0474 |
| Context | Ad hoc document retrieval | RWS | 24s 348ms | 0.0869 | 0.0245 | 0.2654 | 0.2784 |
| | Ad hoc entity retrieval | | 27s 620ms | 0.1304 | 0.0002 | 0.2364 | 0.2298 |
| | Entity list completion | | 19s 422ms | 0.0875 | 0.0373 | 0.0692 | 0.0520 |
| TF-Bins$_2$ | Ad hoc document retrieval | RWS | 2m 58s | 0.0172 | 0.0033 | 0.0500 | 0.0508 |
| | Ad hoc entity retrieval | | 4m 41s | 0.0300 | 0.0000 | 0.1145 | 0.1307 |
| | Entity list completion | | 1m 08s | 0.0006 | 0.0000 | 0.0058 | 0.0053 |
| Syns+Cont. | Ad hoc document retrieval | RWS | 23s 265ms | 0.0882 | 0.0246 | **0.2692** | **0.28830** |
| | Ad hoc entity retrieval | | 26s 877ms | 0.1313 | 0.0002 | **0.2509** | 0.2422 |
| | Entity list completion | | 19s 824ms | **0.0884** | 0.0369 | 0.0788 | 0.0594 |

terms according to word2vec embeddings; the *TF-bins*$_2$ model, which added *tf_bin* hyperedges for the most and least frequent terms per document; and the *Syns+Cont.* model, which included the hyperedges from the *Syns* and *Context* models.

Indexing took between 33h05m (*Syns*) and 37h16m (*Syns+Cont.*) to index on a virtual machine with a 4-core CPU and 32 GB of RAM. The baselines were supported on two Lucene indexes. The first was based on a text-only representation, using the extracted keywords, and it only took 15h06m to index. The second was based on an entity profile, built on the keywords extracted from a virtual document created from the concatenation of sentences mentioning the entity, which took 59h17m to index. This approach has been documented in the literature, notably in the work by Bautin and Skiena [4]. We call these two Lucene indexes *Document Index* and *Entity Index*.

For the ad hoc entity retrieval task over the *Entity Index*, entities were ranked based on the keyword query issued over the Lucene virtual document. For the entity list completion over the *Entity Index*, entities were ranked based on an entity query, by first retrieving the virtual document for each entity in the query, and building a concatenated entity profile. This was then used to issue a `MoreLikeThis` query over the Lucene index, based on the concatenated entity profiles that were loaded through a `StringReader`. We did not impose a minimum term or document frequency, since keywords do not repeat and should, by definition, result in a low document frequency (and thus a high IDF). We also relied on the default value of 25 top keywords according to TF-IDF (or, in practice, IDF, since $TF = 1$) to build the query responsible for retrieving the ranked entities with a similar profile to the input entity set.

Table 9.12 illustrates the performance of each of the three evaluated tasks, over two Lucene and five hypergraph-of-entity representation models. With generalization in mind, our goal is to understand whether the universal ranking function that we propose is able to adequately provide answers for all the tasks, over a common index. This means that we do not necessarily expect performance improvements, but rather an indication that our representation and retrieval model has the potential to be iterated over and improved as a general solution for entity-oriented search (and eventually retrieval).

We provide two ranking function baselines based on Lucene TF-IDF and BM25, which are directly comparable with the random walk score for the same task on the hypergraph-of-entity models. While different indexes and retrieval strategies are required for each task when using Lucene, a single index and ranking function is sufficient to support the three tasks in the hypergraph-of-entity. As we can see, when using document or entity profiles based on the top 5% keywords, the hypergraph-of-entity is able to outperform both Lucene TF-IDF and BM25 in every effectiveness metric, except P@10 and NDCG@10 for the entity list completion task. However, our model is considerably less efficient than Lucene, taking on average from 19s162ms to 4m21s to answer a query, when compared to only milliseconds with Lucene.

A longitudinal view over the three tasks per index, shows a consistent effectiveness for Lucene indexes, with slight variations of the evaluation metrics. However, for hypergraph-of-entity indexes, with the exception of TF-bins$_2$, ad hoc document retrieval and entity list completion showed a consistent effectiveness according to GMAP, particularly within the same representation model, while ad hoc entity retrieval overall resulted in the lowest GMAP scores, but the highest MAP scores, indicating the presence of a small subset of queries with high average precision outlier scores. On the other hand, when focusing on the evaluation metrics over a cutoff of 10, we obtain the best results for ad hoc document retrieval and ad hoc entity retrieval, showing that entity list completion retrieves a similar number of relevant results when compared to ad hoc document retrieval, but they are positioned lower than the top 10 in the rank.

Overall, evaluation scores are low, possibly due to the limitations introduced when considering only the top 5% keywords, however these values for the random walk score are up to par with (and in fact outperform) TF-IDF and BM25 in the same conditions. We assessed the statistical significance of the best model per task when compared with the respective BM25 baseline. Given the non-normality of the average precision scores, we relied on the Wilcoxon signed-rank test. We found that that the *Syns* model was significantly better than *BM25* in ad hoc document retrieval ($p < 0.05$). The *Base Model* was significantly better than *BM25* in ad hoc entity retrieval ($p < 0.05$). The *Syns+Cont.* was significantly better than *BM25* in entity list completion ($p = 0.057$).

We further compared the hypergraph-of-entity MAP score, for each of the three tasks, with the MAP and xinfAP [367] scores from the INEX 2010 Ad Hoc track, and the INEX 2009 XER track. Since our model frequently ranked quite below the state-of-the-art, in performance, for previous experiments, we compared it with the lowest entries from participants in INEX. In the INEX 2010 Ad Hoc track [248], the lowest MAP was 0.3177, compared to 0.0937 for our best model. In the INEX 2009 XML Entity Ranking track [247], the lowest xinfAP for entity ranking was 0.082, compared to a MAP of 0.1390 for our best model, and the lowest xinfAP for entity list completion was 0.100, compared to a MAP of 0.0884 for our best model.

Overall, the experiments we carried, based on a joint representation model of corpora and knowledge bases, and a universal ranking function based on random walks, support the first part of this thesis, stating that:

**Thesis statement (part 1)**

A graph-based joint representation of unstructured and structured data has the potential to unlock novel ranking strategies, that are, in turn, able to support the generalization of entity-oriented search tasks [...]

We were in fact able to propose a general retrieval model for entity-oriented search that is viable, despite its low performance. On the other hand, our results only partially support the second part of this thesis, stating that:

**Thesis statement (part 2)**

[...] and to improve overall retrieval effectiveness by incorporating explicit and implicit information derived from the relations between text found in corpora and entities found in knowledge bases.

While we were able to outperform the Lucene baselines in a particular scenario where documents were represented by a reduced set of representative keywords, we were unable to even approximate the state-of-the-art retrieval approaches described in the literature, specifically when considering the overview reports for the most recent INEX occurrences covering each task.

## 9.4 A REFLECTION ON RETRIEVAL HEURISTICS

We reflect on the presence and effect of information retrieval heuristics in the models we proposed, based on the three foundational concepts used in search: Term Frequency (TF), Inverse Document Frequency (IDF), and Pivoted Document Length Normalization (PDLN). These concepts have been introduced and surveyed in Section 2.1.1. In this section, we analyze how they were considered in the design of graph-based entity-oriented search, in particular in the context of the hypergraph-of-entity, establishing a parallel with BM25.

### 9.4.1 The pillar concepts of information retrieval

The pillars of information retrieval consist of three well-proven concepts, that we briefly revise and comment on next:

- **Term frequency** — The number of times a query term is found in a document should contribute to its relevance. The absolute term count usually suffers a transformation that is a part of the ranking function (e.g., using log or square-root normalization like TF-IDF, or even applying a stronger dampening factor $k_1$ like the one used BM25).
- **Inverse document frequency** — If the term used in the query is too commonly found in the documents of the collection, then it won't be a good discriminator of relevance. Using the term's document frequency establishes an inverse relation with relevance. Similarly to TF, this relation can also be controlled through different transformations.
- **Pivoted document length normalization** — A collection usually contains documents with a diverse number of words. Longer documents usually repeat multiple terms, but also have a higher number of distinct terms, making them more likely to rank higher in general. Normalizing documents based on the average document length (the pivot) mitigates this issue. See Singhal et al. [11, Figure 2] to better understand how PDLN is an approximation of the probability of retrieval and the probability of relevance, plotted over document length, by rotating over the pivot, where the probabilities cross.

### 9.4.2 Deconstructing BM25

In this section, for the sake of discussion and context, we deconstruct BM25, departing from TF-IDF and several heuristics to alter term frequency, until we are able to reconstruct BM25. This is an exercise to demonstrate how the core concepts of a search engine's ranking function, defined in a basic TF-IDF (i.e., without normalizing term count), can be extended to reach a different ranking function, in this case the probabilistic approach given by BM25.

Let us now identify the core components in BM25 [31], starting from Equation 9.1 and building on the analysis work already carried by Rousseau and Vazirgiannis [16, §5], as well as based on some observations by Turnbull [368]:

$$BM25(t,d) = \frac{(k_1+1)\times tf(t,d)}{k_1 \times \left(1-b+b\times\frac{|d|}{avdl}\right)+tf(t,d)} \times \log\frac{N+1}{df(t)} \tag{9.1}$$

Here, $tf(t,d)$ is the frequency of term $t$ in document $d$, $N$ is the number of documents in the collection, and $df(t)$ is the document frequency of term $t$. Before describing parameters $|d|$, $avdl$, $k_1$ and $b$, let us first identify $idf(t,D) = \log\frac{N+1}{df(t)}$ as the inverse document frequency of term $t$ in collection $D$, so that we can build up BM25 from a simple TF-IDF, as described in Equation 9.2.

$$TF\text{-}IDF(t,d) = tf(t,d)\times\log\frac{N+1}{df(t)} \tag{9.2}$$

By taking a probabilistic view, and departing from the three pillars of information retrieval, our goal should be to approximate the probability of retrieval and the probability of relevance, not only over document length, but also over term frequency and inverse document frequency. Were the probability of retrieval to perfectly match the probability of relevance and perfect a ranking could be obtained.

First, let us detail the component controlled by $b$, which is used to regulate the degree of pivoted document length normalization affecting term frequency. Here, $|d|$ is the document length in number of words, while $avdl$ is the average document length for the collection:

$$tf'(t,d) = \frac{tf(t,d)}{1-b+b\times\frac{|d|}{avdl}} \tag{9.3}$$

Now, let us detail the component controller by $k_1$, which is used to establish an upper bound of $k_1+1$ for term frequency as term count increases. This parameter is usually set to $k_1 = 1.2$ or optimized in the interval $k_1 \in [1.2, 2]$. The logic behind this decision is that, after a certain number of repetitions of a given term, there are no repetitions that will increase relevance. Let us then rewrite term frequency to approach $k_1+1$ asymptotically as term count increases:

$$tf''(t,d) = \frac{(k_1+1)\times tf'(t,d)}{k_1+tf'(t,d)} \tag{9.4}$$

In Equation 9.5, we combine the two term frequency approaches by expanding $tf'(t,d)$ in Equation 9.4, while using $K = 1 - b + b \times \frac{|d|}{avdl}$ for convenience:

$$\begin{aligned} tf''(t,d) &= \frac{(k_1+1) \times tf'(t,d)}{k_1 + tf'(t,d)} & &= \frac{(k_1+1) \times \frac{tf(t,d)}{K}}{k_1 + \frac{tf(t,d)}{K}} \\ &= \frac{(k_1+1) \times \frac{tf(t,d)}{\cancel{K}} \times \cancel{K}}{\left(k_1 + \frac{tf(t,d)}{K}\right) \times K} & &= \frac{(k_1+1) \times tf(t,d)}{k_1 \times K + \frac{tf(t,d) \times \cancel{K}}{\cancel{K}}} \\ &= \frac{(k_1+1) \times tf(t,d)}{k_1 \times K + tf(t,d)} & &= \frac{(k_1+1) \times tf(t,d)}{k_1 \times \left(1 - b + b \times \frac{|d|}{avdl}\right) + tf(t,d)} \end{aligned} \tag{9.5}$$

Interestingly, despite relying on different normalization approaches, IDF is rarely reiterated over in ranking models, conceptually still following the original proposal by Spärck Jones [9]. By combining IDF with the described term frequency transformation, we obtain the well-known BM25.

We went through the exercise of illustrating BM25's components and how different interpretations of term frequency's behavior result in different ranking functions. Some of them, like BM25, represent an improvement over TF-IDF, as they rely on a bound term frequency that is normalized for document length.

### 9.4.3 Deconstructing the hypergraph-of-entity

The three pillars, TF, IDF and PDLN, are always present, even if only conceptually, in a good ranking function. This has been proven by years of research in information retrieval and multiple successful ranking models that rely on these core principles (TF-IDF, BM25, language models, divergence from randomness, Bayesian networks, Markov networks; see Section 2.1.1).

Many of our experiments with the hypergraph-of-entity obtained MAP scores lower than the TF-IDF and BM25 baselines. This might be attributed to missing implementations or analogies of one or multiple of the core principles. The hypergraph-of-entity can be seen as a model for representation-driven retrieval, since the random walk score is essentially a biased sampler of the hypergraph, that depends on the query and the internal weights of the nodes and hyperedges. As such, there might be a redundancy in relying on inverse document frequency or document length normalization. That is, these principles might already be a part of the model, inherently. Let us delve deeper into this possibility.

**TERM FREQUENCY** While we have explored the inclusion of term frequency in our model, through TF-bins, there is still much work that can be done to explore the optimal approach. For instance, the weight of *tf_bin* hyperedges was not computed based on the term frequencies of its terms in the documents they belong to. Is there a way to do this in a way that improves performance? Since the remaining weights of the hypergraph are normalized in the range $[0, 1]$, how could we compute a TF-bin weight that would be on the same scale, as an indicator of relevance? We might even need to go through a similar process that Singhal et al. [11] went through to propose pivoted document length normalization, studying the probability of retrieval based on different weighting schemes and comparing it with the probability of relevance.

**INVERSE DOCUMENT FREQUENCY** When a term is particularly prominent in a collection of documents, it becomes less discriminative. In the hypergraph-of-entity, there is no equivalent computation based on this concept. Since terms with a high IDF are not filtered out, random walks can freely traverse through these terms from document to document, inadvertently assigning relevance to otherwise irrelevant

documents. Although we aggressively remove stopwords and potentially irrelevant terms with a low length, there is still the possibility of domain-specific terms being used in a collection, therefore remaining a part of the hypergraph-of-entity. It is a possibility that, by removing terms with a high IDF prior to adding documents to the hypergraph, we might improve retrieval effectiveness.

**DOCUMENT LENGTH NORMALIZATION** Two aspects were identified to justify the need for pivoted document length normalization: higher term frequencies, and a higher number of diverse terms, in long documents. The first issue is inherently solved in the hypergraph-of-entity, due to the lack of term repetition (documents are represented as an undirected hyperedge, which is a set of terms and entities). On the other hand, the second issue prevails, as long documents result in higher cardinality *document* hyperedges. During manual trials, we frequently found that longer documents were given a higher score than shorter, sometimes more relevant, documents. This is an issue that must be addressed in the future, preferably by modifying the structure of the hypergraph-of-entity to accommodate this concept organically. This effort will ensure that the hypergraph remains algebraically representable, perhaps as a tensor, which might lead to a significant improve in efficiency, should we take advantage of a Multilinear PageRank with personalization to answer queries.

## SUMMARY

In this chapter, we evaluated the hypergraph-of-entity representation and retrieval model. First, we focused on exploring different configurations of the representation model, for a single task (ad hoc document retrieval). We ran several experiments over subsets for INEX 2009 Wikipedia collection, and over TREC Washington Post Corpus based on our participation in TREC 2018 Common Core track. This included statistics about the hypergraph used to index the collections, a study of rank stability, the assessment of retrieval performance for the ad hoc document retrieval task, and a comparison with the graph-of-entity. Then, we ran multiple experiments to test the random walk score as a universal ranking function, by validating whether it supported multiple ranking tasks over a single index, based on the hypergraph-of-entity representation model for the complete INEX 2009 Wikipedia collection. We described an approach to evaluate general retrieval models, and proposed a strategy to scale the hypergraph-of-entity based on keyword extraction, which enabled us to compare results with the literature for the INEX tracks. We successfully proved the first part of our thesis statement, proposing that a graph-based general model for entity-oriented search would be possible, and we partially proved the second part, showing that, in some conditions, we were able to outperform the Lucene baselines by cross-referencing information from corpora and knowledge bases. We closed with a discussion on the pillar concepts of information retrieval, covering their presence or absence in BM25 and repeating this analysis for the hypergraph-of-entity.

# Part IV

# CONCLUSION

# 10 DISCUSSION

## Contents



Information retrieval experts have relied on different solutions to solve individual tasks required to bring search to the user, when starting from a corpus and/or knowledge base. Search is viewed as a set of multiple problems and multiple solutions, but there is fewer work on unified approaches or joint models. In this thesis, we have tackled multiple entity-oriented search problems through a single solution, with the disadvantage of lacking specialization at each task, but also with the advantage of harnessing all available information so that, through a single process, we were able to solve any information need, regardless of the retrieval task that was used to express it. In the hypergraph-of-entity, any element can be ranked or used to query.

We have experienced the process of designing a novel representation and retrieval model nearly from zero and, while it partly integrates the pillar concepts of information retrieval, it also defies the *status quo* of the area, by deviating from the inverted file and the triplestore. In this process, we proposed a model for joint representation of corpora and knowledge bases, thus promoting unified approaches for entity-oriented search based on a common data structure. Without combining text, entities and their relations we would might miss crucial connections between concepts (see the astronaut and entertainer example in Section 7.1).

We proposed two main models, graph-of-entity (Chapter 6) and hypergraph-of-entity (Chapters 7, 8 and 9), as well other adjacent graph-based concepts, such as fatigued random walks (Appendix B), that were tested as performance enhancement elements. In this chapter, we present an overview on the experimental process and the obtained results per collection, discussing the limitations and potential applications of the hypergraph-of-entity as a general model for solving information needs, while promoting the exploration of novel views in information retrieval.

The structure of this chapter is organized as follows:

- **Section 10.1** reflects on the five stages of experimentation undergone while designing the hypergraph-of-entity, from conception to generalization testing.
- **Section 10.2** provides a global view of the carried scientific work, including a timeline of experiments and hypergraph-of-entity structural feature changes. We classify experiments according to the used test collection, so that we can compare them. In particular, we cover experiments over SSOAR [§10.2.1], INEX 2009 10T-NL [§10.2.2], INEX 2009 52T-NL [§10.2.3], WaPo [§10.2.4], and the complete INEX 2009 Wikipedia collection [§10.2.5].
- **Section 10.3** discusses the limitations of the hypergraph-of-entity, covering the issues with the document preprocessing pipeline [§10.3.1], keyword-based profiles [§10.3.2], the random walk score [§10.3.3], and the usage of hypergraphs as a data structure for the generalization of retrieval [§10.3.4].

## 10.1 FIVE STAGES OF EXPERIMENTATION

A great part of this doctoral work can be summarized by instancing and describing five stages of experimentation (not necessarily in temporal order of execution) leading to the conclusion of this thesis:

- Conception
- Representation model
- Retrieval model
- Fallback to classical
- Generalization testing

**CONCEPTION STAGE** At the conception stage, we were concerned with establishing a foundation for our work. We did this by exploring the graph-of-word [16] and, inspired by this model, we attempted our own graph-of-entity model (Chapter 6). The graph-of-entity deviated quite a bit from the graph-of-word, for multiple reasons. The graph-of-word is a document-based model, which means that a graph is created per document, enabling metrics to be extracted from the graph, as an offline process, and even stored in an inverted index. On the other hand, the graph-of-entity was designed to be a collection-based model, as per our goal of harnessing "all available information". This meant that it would not be viable to maintain the sliding-window-based contextual connections from graph-of-word, as this would result in a computationally intractable representation. Our first decision was then to reduce to window size to one term, which in practice meant that each term only connected to its following term. Another reason to do this was the addition of semantic information, through the introduction of entities in the graph. This also meant that the model would grow in complexity, not only because of the added entities, but also because of the added relations, both between pairs of entities and between terms and entities. These were strong reasons to preemptively reduce the size of the graph. This is when we started to think about approaches that would enable us to further reduce the size of the model, which led us to hypergraphs (see Figure 9.2). The first advantage we identified in hypergraphs was the fact that we would be able to group several nodes in a single hyperedge and we could test whether such simplification, along with an increase in the number of documents

and entities, would lead to better results. At this point, we experimented with a hypergraph database[1], but rapidly moved to an in-memory representation for further increased efficiency[2].

**REPRESENTATION MODEL STAGE** The representation model stage was marked by the introduction of the hypergraph–of–entity (Chapter 7), which was our main focus throughout this doctoral work. We first designed a simplified hypergraph-based version of the graph-of-entity, while simultaneously exploring alternative, more efficient approaches to ranking. Inspired by the entity weight, we proposed the random walk score, which, instead of relying on all simple paths departing from seed nodes, relied on a sample of those paths, given by random walks. Once this framework was established, we focused on extending the representation model and assessing the effectiveness and efficiency for several variations of the model. We explored a text-only version, and versions relying both on internal and external knowledge, as well as synonyms, contextual similarity, TF-bins, weights, and several combinations of these features, while maintaining a static configuration for the random walk score. We also characterized the structure of the hypergraph for many of these representation model variations, while attempting to understand the impact of structural features in retrieval performance.

**RETRIEVAL MODEL STAGE** At the retrieval model stage, we turned our focus to the parameterization of the ranking function, while also characterizing rank stability, given the nondeterminism of random walks. This mostly happened while also exploring the variations of the representation model, although we focused on most descriptive or overall best results to present. An exception was the node and edge fatigue ($\Delta_{nf}$ and $\Delta_{ef}$), which we explored and presented in a more exhaustive manner. Inspired by neuronal fatigue, a phenomenon of the brain, we attempted to block node and hyperedge visitation during a given $\Delta_{nf}$ or $\Delta_{ef}$ cycles after being visited. This was a failed attempt to improve efficiency without decreasing effectiveness. In fact, it had the opposite effect in most cases. Despite the low gain rarely provided by fatigue, we decided to further explore and develop this concept in order to exhaust the idea. One idea we had was the translation of our step-by-step random walk simulation to an algebraic simulation supported on Markov chains and power iteration. This required a tensor-based representation of the hypergraph-of-entity, which is not only a general hypergraph, but also a mixed hypergraph, i.e., with directed and undirected hyperedges. While we found significant advances that might support this approach for an undirected hypergraph [115, 201], the challenge to apply them was significant, but it also would not result in a perfect fit for our model. As such, we decided to test the idea of fatigue in a more controlled graph-based scenario instead, and only then come back to hypergraphs, if it first showed promise there.

**FALLBACK TO CLASSICAL** At this stage, we proposed Fatigued PageRank as a classical link analysis metric, based on an approximated implementation of fatigued random walks, represented as a Markov chain and solvable through power iteration. We then compared it with several other graph-based features, computed over a link graph, measuring their impact in effectiveness over a text-based score. In particular, we experimented with indegree, HITS authority, and PageRank, as well as Fatigued PageRank. While the indegree had a negative impact overall and a positive impact for the top 10 results, the remaining metrics had close to no impact overall. As such, we moved on to the next stage of experimentation, leaving fatigue on hold as future work.

1 http://hypergraphdb.org/
2 http://www.i3s.unice.fr/~hogie/software/index.php?name=grph

**Table 10.1:** Summary of the experimental stages covered by different test collection.

| Stages \ Sections | SSOAR §10.2.1 | 10T-NL §10.2.2 | 52T-NL §10.2.3 | WaPo §10.2.4 | INEX §10.2.5 |
|---|---|---|---|---|---|
| **Conception** | ✓ | ✓ | | | |
| **Representation model** | | ✓ | ✓ | ✓ | |
| **Retrieval model** | | ✓ | | | |
| **Fallback to classical** | | | | ✓ | |
| **Generalization testing** | | | | | ✓ |

**GENERALIZATION TESTING** Perhaps the most central point of this thesis was the generalization of entity-oriented search tasks. In particular, we decided to focus on graph-based models to represent corpora and knowledge bases and to support ranking. The generalization testing stage was about assessing the performance of three different tasks: ad hoc document retrieval, ad hoc entity retrieval, and entity list completion. We did this based on the complete INEX 2009 Wikipedia collection, representing each document by its top keywords instead of the full text, in order to reduce computational complexity. Under these specific conditions, we are able to outperform both Lucene TF-IDF and BM25 baselines. However, based on INEX 2009 10T-NL, we also found that no hypergraph-of-entity based on extracted keywords could outperform baselines that relied on the full text.

## 10.2 A GLOBAL VIEW OF RESULTS PER TEST COLLECTION

In the following sections, we comment on the overall results obtained per test collection, for different experiments. We do this based on summary tables of performance metrics, as well as by relying on plots of the relative change according to the baselines defined for each specific experiment.

Table 10.1 shows how the six stages of experimentation were distributed over the different test collections. In Section 10.2.1, we focus solely on the conception stage, with the first proposal of graph-of-entity, tested over SSOAR using team-draft interleaving, during the TREC 2017 OpenSearch track. Section 10.2.2 covers all experiments over the INEX 2009 10T-NL Wikipedia subset, which re-tests graph-of-entity and introduces hypergraph-of-entity (conception stage). In that section, we also explore several variations of the hypergraph-of-entity (representation model stage), as well as several parameterizations for the random walk score (retrieval model stage). In Section 10.2.3, we mainly focus on the representation model stage, over the INEX 2009 52T-NL Wikipedia subset. In Section 10.2.4, we study a stripped down representation model of the hypergraph-of-entity (text-only), and we also enter the fallback to classical stage, exploring classical link analysis approaches to improve text-based ranking functions. We do this over TREC Washington Post Corpus (WaPo), during TREC 2018 Common Core track. Finally, in Section 10.2.5, we run our final experiments over INEX 2009 Wikipedia collection, where we explore generalization by testing multiple entity-oriented search tasks based on a single representation and retrieval model.

As experiments were executed, several structural changes were explored, particularly during the design of the hypergraph-of-entity. In order to make it clear which variation of this structure was considered for each experiment, we provide in Figure 10.1 a timeline with the feature changes at the top and a summary of the carried experiments at the bottom. The main experimental stage ranged from July 2017, with the participation in TREC 2017 OpenSearch track, to the present (May 2020), with the final generalization experiments over the hypergraph-of-entity. Initially, we had considered the introduction of *document* nodes in the hypergraph-of-entity, along with directed *document* hyperedges, however it became clear that, in particular for longer documents, the probability of a random walk visiting the

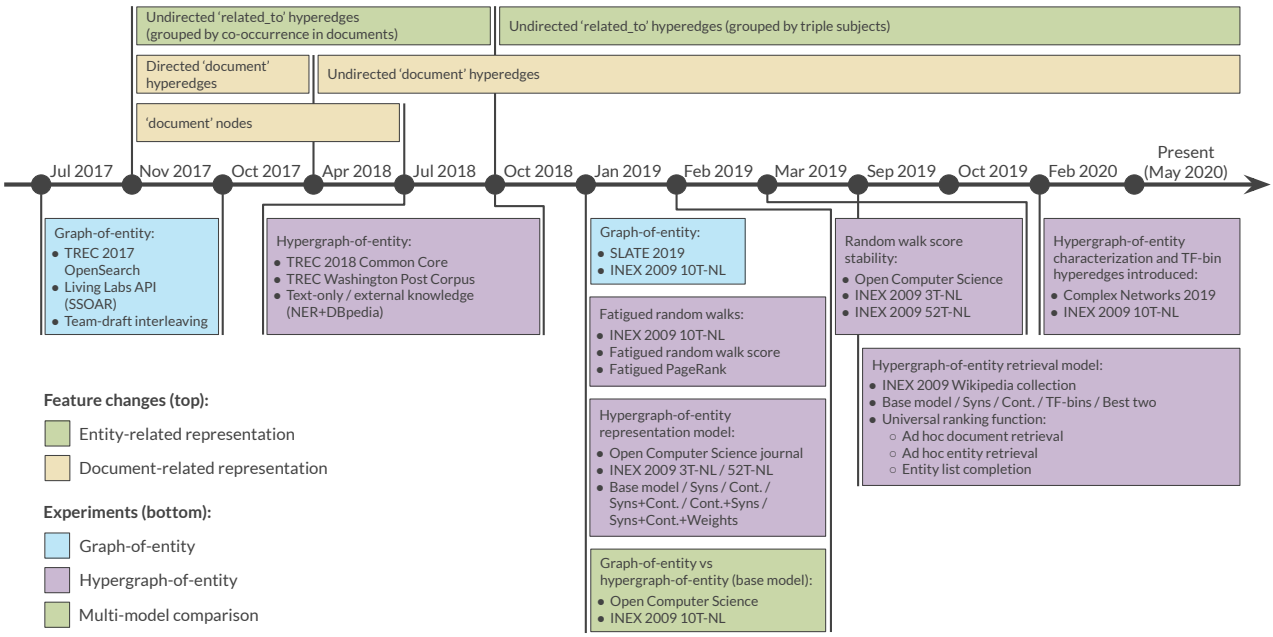



**Figure 10.1:** Experimentation timeline (bottom) and evolution of hypergraph-of-entity model features (top).

**Table 10.2:** Overall comparison of retrieval performance, for the ad hoc document retrieval task, based on the TREC 2017 OpenSearch track SSOAR dataset.

| Index | Ranking | Win Ratio | Index Time | Avg./Query | Nodes | Edges |
|---|---|---|---|---|---|---|
| GoW | TW-IDF | $6/(6+10) = 0.375$ | 31m 59s | – | 201,856 | 9,530,421 |
| GoE | EW | $2/(2+10) = 0.167$ | 49m 29s | – | 261,215 | 3,482,308 |

document node was minimal, despite the walker frequently having visited nodes from that same document. Thus, in July 2018 we dropped *document* nodes, working only with undirected *document* hyperedges from then on. All experiments over the hypergraph-of-entity that are reported on this thesis were executed over a model with these structural conditions. Regarding the *related_to* hyperedge variations, only the TREC 2018 Common Core track experiments relied on the co-occurrence of entities over a common document to form a hyperedge, while the remaining experiments relied on *related_to* hyperedges grouping entities connected to a common triple subject (the subject was also included in the set).

### 10.2.1 SSOAR via Living Labs

Our initial experiments, during the conceptual stage, were focused on assessing a collection-based implementation of graph-of-word, as well as the graph-of-entity model that we proposed. We did this based on team-draft interleaving, through the Living Labs platform at TREC 2017 OpenSearch track, over the Social Science Open Access Repository (SSOAR). The evaluation result was computed as an outcome metric based on the fraction of wins of our search engine over the original search engine (Lucene TF-IDF). An outcome of 50% would represent an equivalent performance, while a superior fraction would represent a better performance for our search engine, or vice-versa. As shown in Table 10.2, we obtained 37% wins for graph-of-word, and 16.7% wins for graph-of-entity, which means that both of these models were outperformed by TF-ID as run in SSOAR, with graph-of-entity being the lowest performing model. This was unexpected, since the literature supported graph-of-word as a better algorithm than TF-IDF, sometimes even outperforming

**Table 10.3:** Overall comparison of retrieval performance, for the ad hoc document retrieval task, based on INEX 2009 10T-NL.

| Index | Ranking | MAP | P@10 | Index Time | Avg./Query | Nodes | Edges |
|---|---|---|---|---|---|---|---|
| Lucene | TF-IDF | 0.1710 | 0.2800 | 27s 769ms | 209ms | – | – |
| | BM25 | 0.2963 | 0.5000 | | 316ms | – | – |
| GoW | TW-IDF | 0.2333 | 0.3000 | 1h 23m | 1m 50s | 492,185 | 22,906,803 |
| GoE | EW | 0.0399 | 0.1500 | 1h 38m | 21s 557ms | 981,647 | 9,942,647 |
| | Fixed parameters: $RWS(\ell = 2, \Delta_{nf} = 0, \Delta_{ef} = 0, expansion = \text{true})$ | | | | | | |
| HGoE (Base Model) | $RWS(r = 10^1)$ | 0.0485 | 0.1200 | 53s 922ms | 943ms | 607,213 | 253,154 |
| | $RWS(r = 10^2)$ | 0.1118 | 0.1500 | | 11s 134ms | | |
| | $RWS(r = 10^3)$ | 0.1492 | 0.2200 | | 1m 18s | | |
| | $RWS(r = 10^4)$ | 0.1689 | 0.1700 | | 13m 04s | | |
| Lucene | TF-IDF | 0.2160 | 0.2800 | 25s 889ms | 451ms | – | – |
| | BM25 | 0.3412 | 0.4900 | | 269ms | – | – |
| | Fixed parameters over HGoE variations: $RWS(\ell = 2, r = 10^3, expansion = \text{true})$ | | | | | | |
| Base Model | $RWS(\Delta_{nf} = 0, \Delta_{ef} = 0)$ | 0.1560 | 0.1800 | 1m 10s | 1m 14s | 607,213 | 253,154 |
| | $RWS(\Delta_{nf} = 0, \Delta_{ef} = 10)$ | 0.1601 | 0.2300 | | 1m 22s | | |
| | $RWS(\Delta_{nf} = 10, \Delta_{ef} = 0)$ | 0.0249 | 0.0900 | | 839ms | | |
| | $RWS(\Delta_{nf} = 10, \Delta_{ef} = 10)$ | 0.0246 | 0.1000 | | 834ms | | |
| Syns +Cont. | $RWS(\Delta_{nf} = 0, \Delta_{ef} = 0)$ | 0.1594 | 0.2300 | 1m 4s | 1m 05s | 699,363 | 422,924 |
| | $RWS(\Delta_{nf} = 0, \Delta_{ef} = 10)$ | 0.1540 | 0.2000 | | 1m 05s | | |
| | $RWS(\Delta_{nf} = 10, \Delta_{ef} = 0)$ | 0.0236 | 0.0900 | | 955ms | | |
| | $RWS(\Delta_{nf} = 10, \Delta_{ef} = 10)$ | 0.0272 | 0.1100 | | 924ms | | |
| Syns +Cont. +Weight | $RWS(\Delta_{nf} = 0, \Delta_{ef} = 0)$ | 0.1636 | 0.2300 | 1m 17s | 5m 26s | 699,363 | 422,924 |
| | $RWS(\Delta_{nf} = 0, \Delta_{ef} = 10)$ | 0.1615 | 0.1900 | | 5m 31s | | |
| | $RWS(\Delta_{nf} = 10, \Delta_{ef} = 0)$ | 0.0195 | 0.0700 | | 1s 011ms | | |
| | $RWS(\Delta_{nf} = 10, \Delta_{ef} = 10)$ | 0.0250 | 0.1200 | | 1s 049ms | | |
| Lucene | TF-IDF | 0.2160 | 0.2800 | 25s 889ms | 451ms | – | – |
| | BM25 | 0.3412 | 0.4900 | | 269ms | – | – |
| | Fixed parameters over HGoE variations: $RWS(\ell = 2, r = 10^4, \Delta_{nf} = 0, \Delta_{ef} = 0, expansion = \text{false})$ | | | | | | |
| Base M. | RWS | 0.0039 | 0.0400 | – | 4s 653ms | 607,213 | 253,154 |
| Syns Context | RWS | 0.0024 | 0.0400 | – | 4s 391ms | 610,212 | 263,804 |
| | RWS | 0.0010 | 0.0100 | – | 4s 179ms | 697,068 | 410,371 |
| $TF\text{-}Bins_2$ | RWS | 0.1025 | 0.2000 | – | 15s 378ms | 607,213 | 268,100 |
| $TF\text{-}Bins_{10}$ | RWS | 0.1133 | 0.1600 | – | 11s 084ms | 607,213 | 281,642 |

BM25. For this reason, we decided to keep testing our models with other datasets. Despite underperforming, graph-of-entity still contained 6 million less edges and only 60 thousand more nodes than graph-of-word.

### 10.2.2 INEX 2009 10T-NL Wikipedia subset

During the conceptual stage of the hypergraph-of-entity, we required a smaller sample of the 2.6 million document INEX 2009 Wikipedia collection. Accordingly we created the INEX 2009 10T-NL subset, that we used in three experiments based on the ad hoc document retrieval task. As we can see on Table 10.3, the first experiment was used to assess the performance of the hypergraph-of-entity in ad hoc document retrieval, when compared to the previously defined graph-of-entity model, as well as the Lucene and graph-of-word baselines. While the hypergraph-based model was able to outperform the graph-of-entity, particularly for higher values of r, it still could not outperform Lucene TF-IDF and BM25. For example, the best MAP for RWS was of 0.1689, which surpassed graph-of-entity with only 0.0399, but did not perform better than TF-IDF which achieved a MAP of 0.1710, or BM25 which achieved a MAP of 0.2963, or even graph-of-word, with a MAP of 0.2333. With this experiment, we also entered the retrieval model stage, by focusing on the impact of the r parameter. Despite the improvements brought by the hypergraph-of-entity, which were still below the baselines, there was a clear trade-off between effectiveness and efficiency — as r increased, so did MAP, but also the average query time.

Efficiency is a relevant problem that, while outside the focus of this thesis, needs to be addressed if this retrieval model is to be used in a production scenario, which it is currently far from achieving.

In the second experiment, described in Table 10.3, we explored several different fatigue parameterizations over several variations of the hypergraph-of-entity representation model (previously introduced, but only covered ahead in Section 10.2.3). This experiment contributed not only to the retrieval model stage, but also to the representation model stage. Here, fatigue was configured according to a delta ($\Delta_{nf}$ and $\Delta_{ef}$) representing the number of cycles where a previously visited node or hyperedge will be fatigued and temporarily not accepting visits — this required that this state was tracked per node and hyperedge, however the impact was negligible. The goal of fatigue was to introduce diversification in the random walk, with the goal of covering a larger portion of the graph. We obtained the best results for the versions without fatigue or relying only on edge fatigue (MAP $\approx$ 0.16), while node fatigue had a strong negative impact on the overall performance (MAP $\approx$ 0.02). Overall, while the idea of implementing a fatigued random walk is interesting, this particular implementation was unsuccessful, and we dropped it for the hypergraph-of-entity after this experiment.

In the third experiment, described in Table 10.3, we introduced a preliminary version of TF-bins, while also exploring results without query expansion, i.e., using the term nodes directly as seed nodes, as opposed to considering the adjacent entity nodes. Not using expansion resulted in lower effectiveness and higher efficiency — however the results for MAP were as low as 0.0010. Regarding TF-bins, we explored different numbers of bins, from 2 to 10 bins, however, in this summary table, we only show the bounds of the interval. TF-bins were assigned weights in accordance with the TF of the terms that they contained, but not directly based on the term frequencies of the set — e.g., for two TF-bins (or TF-bins$_2$), the hyperedge representing the bin of terms with highest TF would have weight 2/2, while the hyperedge representing the bin of terms with lowest TF would have weight 1/2. While TF-bins are inherently weighted hyperedges, in this preliminary version, we had not computed weights for the remaining nodes and hyperedges, which resulted in a biased random walk that highly prioritized *tf_bin* hyperedges, selecting the remaining nodes and hyperedges with infinitesimal uniform probability, in practice meaning that only *tf_bin* hyperedges were traversed, ignoring the remaining hyperedges of the model. In short, the model was used as if it were a text-only version containing only terms and *tf_bin* hyperedges. Although we achieved the best scores for this implementation of TF-bins (MAP $\approx$ 0.10), it was still below the results obtained in the first experiment for $r = 10^4$ (MAP $\approx$ 0.17), and indicator that the remaining structure of the hypergraph-of-entity is relevant for effectiveness.

Despite relying on different pre-processing pipelines (the reason why we rely on separate BM25 baselines per experiment), it is clear that graph-of-entity produces the highest number of nodes (nearly 1 million) and graph-of-word produces the highest number of edges (nearly 23 million), although the model was not designed to be implemented as a collection-based graph, but rather as a document-based graph. On the other hand, the hypergraph-of-entity ranges between 600 to 700 thousand nodes, but even the largest hypergraph only requires a little under half-million hyperedges, with the base model or the synonyms model requiring a little over 200 thousand hyperedges. This is a clear advantage of hyperedges, benefiting not only storage requirements, but also retrieval efficiency, enabling for more features to be considered. It is, however, relevant to highlight the fact that efficiency-wise the our model is still quite below the state-of-the-art.

Figure 10.2 provides an overview of the relative change in MAP and P@10 for all runs launched over INEX 2009 10T-NL. Each run is compared with its own BM25 baseline (not displayed in the figure). Different representation models are shown in different colors, with the ranking function and the most relevant parameters shown in the y-axis, and the change in MAP or P@10 shown in the x-axis. As we can see,

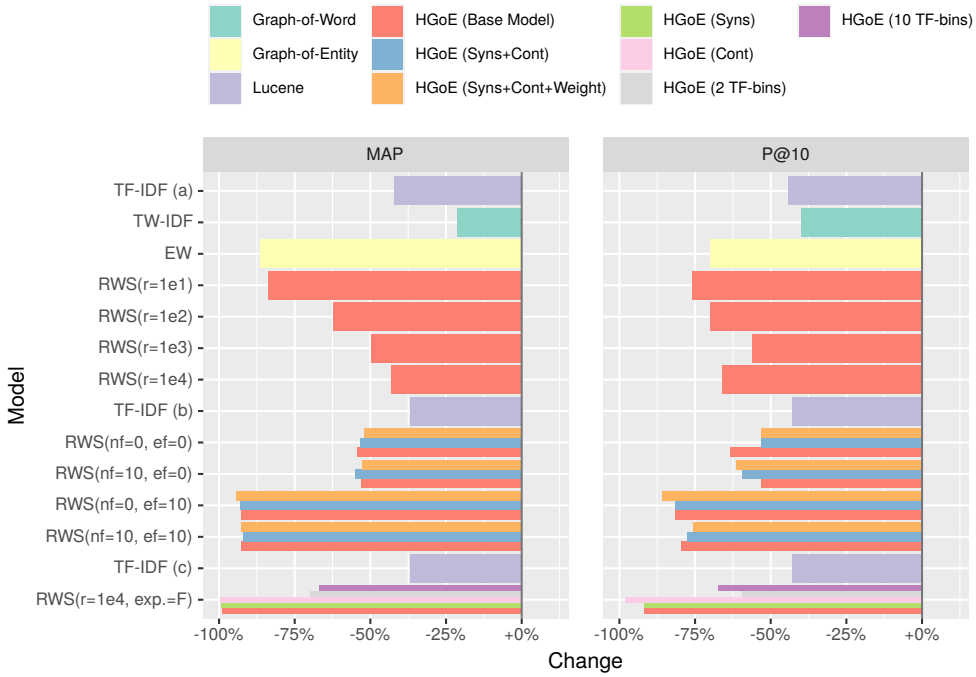



**Figure 10.2:** Relative change in MAP and P@10 compared to the BM25 baseline for each subset of experiments over INEX 2009 10T-NL.

**Table 10.4:** Overall comparison of retrieval performance, for the ad hoc document retrieval task, based on INEX 2009 52T-NL.

| Index | Ranking | MAP | P@10 | Index Time | Avg./Query | Nodes | Edges |
|---|---|---|---|---|---|---|---|
| Lucene | TF-IDF | 0.1689 | 0.2346 | 1m 21s | 1s 148ms | – | – |
| | BM25 | 0.3269 | 0.5250 | | 1s 220ms | – | – |
| Fixed parameters over HGoE variations: $RWS(\ell = 2, r = 10^3, \Delta_{nf} = 0, \Delta_{ef} = 0, expansion = \text{true})$ | | | | | | | |
| Base Model | RWS | 0.0864 | 0.1269 | 4m 06s | 3m 23s | – | – |
| Syns | | 0.0840 | 0.1231 | 3m 55s | 3m 31s | – | – |
| Cont. | | 0.0811 | 0.1192 | 3m 59s | 3m 36s | – | – |
| Syns+Cont. | | 0.0837 | 0.1231 | 3m 57s | 3m 33s | – | – |
| Cont.+Syns | | 0.0814 | 0.1250 | 3m 59s | 3m 36s | – | – |
| Syns+Cont.+Weight | | 0.0884 | 0.1154 | 4m 06s | 10m 56s | 2,031,848 | 1,142,309 |

none of the ranking functions was able to was able to outperform the BM25 baseline, including TF-IDF and TW-IDF from the graph-of-word. From the MAP plot, it is clearly visible the impact of that a high number of iterations r have in the retrieval effectiveness, as RWS approximates TF-IDF for $r = 10^4$. This same impact is still visible in P@10, although with less intensity. Experiments without query expansion suffered the most negative change in regard to BM25.

### 10.2.3 INEX 2009 52T-NL Wikipedia subset

The experiment shown in Table 10.4 can be positioned in the representation model stage, relying on an extended subset built from the complete set of 52 topics provided for INEX 2010 Ad Hoc track. Like other subsets, efficiency was ensured by limiting the collection to the documents mentioned in the relevance judgments for the 2010 topics. At this stage, we hadn't yet proposed TF-bins. Otherwise, we tested a representative set of combinations for all the available hypergraph-of-entity extensions. This included exploring synonyms, as well as contextually similar terms, which can be used for query expansion. However, in the hypergraph-of-entity, they are explicitly defined in the index as a part of the representation model. We also ex-

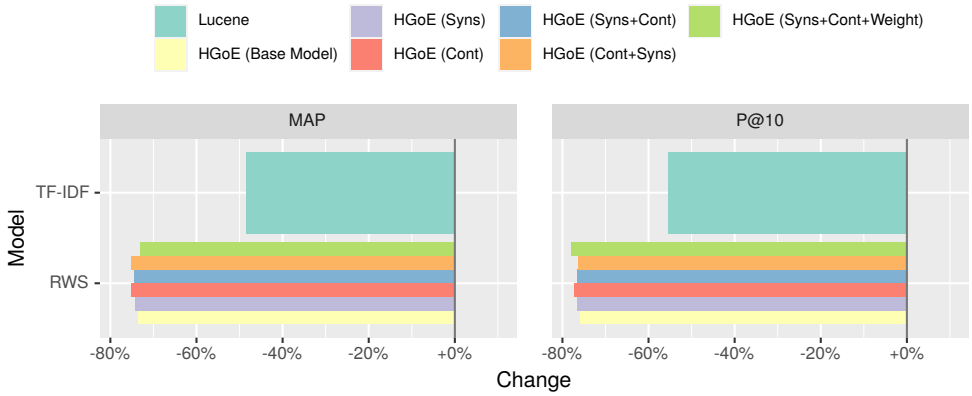



**Figure 10.3:** Relative change in MAP and P@10 compared to the BM25 baseline for each subset of experiments over INEX 2009 52T-NL.

plored the possibility of introducing bias through a simply weighting scheme that provided additional information per node or hyperedge type.

It is frequent for synonyms and contextually similar terms to overlap — in Figure 7.5 from Chapter 7, we had found the term *results* to be both contextually similar and a synonym of the term *outcome*; the same happened for *intent* and *intention*. In such cases of overlap, each extension reinforces the probability of visiting that term during the computation of the random walk score. It was particularly interesting to find that both synonyms and contextually similar terms were complementary. Moreover, the order by which we added synonyms and context to the index makes a difference. For the *Syns+Cont.* version, synonyms are added over the term vocabulary of the original collection, and then contextually similar terms are added for both the original terms and their synonyms. In a similar way, the opposite also happens for the *Cont.+Syns* version.

We created an index for both combinations. However, in this particular case, since the word2vec model was trained based on the same INEX 2009 52T-NL subset, they are in fact equivalent — i.e., even after adding new terms from synonyms, these won't be present in the word2vec model, and, conversely, after adding context, no new terms will be added, since the word2vec model was trained on the same collection, and therefore no new terms from synonyms will be added. However, depending on the train set used for word2vec, this will not always be the case, so we included both versions at this stage. Additionally, given the nondeterministic nature of the random walk score, it also serves to illustrate the small variations in evaluation scores.

Regarding the actual results, and despite using expansion (i.e., seed nodes based on terms expanded to adjacent entity nodes), which had obtained the best results over INEX 2009 10T-NL, we obtained MAP and P@10 scores that were much lower than the TF-IDF and BM25 baselines. For example, we obtained a MAP of 0.17 for TF-IDF, compared to a MAP of 0.09 for RWS over the *Syns+Cont.+Weight* hypergraph-of-entity. Additionally, for this particular subset, there was little variation over the MAP and P@10 scores for different versions of the hypergraph-of-entity. The index time and average query time was also similar for all variations, however, the weighted approach resulted in a significantly higher average query time, due to the inefficiency introduced by the biased random walk implementation. Regarding hypergraph size, the number of edges was kept at half the number of nodes, as measured for the *Syns+Cont.+Weight* version[1].

1 We were unable to measure the number of nodes and edges for the remaining hypergraph-of-entity versions, since the index inspection method had not been implemented at the time we ran the experiments using unbiased random walks.

**Table 10.5:** Overall comparison of retrieval performance, for the ad hoc document retrieval task, based on TREC Washington Post Corpus.

| Index | Ranking | MAP | P@10 | Index Time | Avg./Query | Nodes | Edges |
|---|---|---|---|---|---|---|---|
| | Fixed parameters: BM25($k_1 = 1.2, b = 0.75$) ⊦ sigm($w = 1.8, k = 1, a = 0.6$) | | | | | | |
| Lucene | BM25 | 0.2031 | 0.3700 | – | 1s 447ms | 159,228 | 319,985 |
| | + Indegree | 0.1994 | 0.3800 | | 1s 537ms | | |
| | + HITS Authority | 0.2031 | 0.3720 | | 1s 462ms | | |
| | + PageRank | 0.2031 | 0.3700 | | 1s 453ms | | |
| | + Fatigued PageRank | 0.2031 | 0.3700 | | 1s 450ms | | |
| | Fixed parameters over HGoE variations: RWS($\ell = 2, r = 10^3, \Delta_{nf} = 0, \Delta_{ef} = 0, expansion = true$) | | | | | | |
| Text-only | RWS | 0.0070 | 0.0680 | 20m 16s | 7s 217ms | 886,298 | 595,037 |
| DBpedia | RWS | 0.0051 | 0.0240 | 29m 16s | 5m 40s | 1,152,850 | 1,447,298 |

Figure 10.3 provides an overview of the change in MAP and P@10 when compared to the BM25 baseline. As we can see, according to MAP, the *Syns+Cont.+Weight* and the *Base Model* are the two best performing versions. On the other hand, according to P@10, the *Syns+Cont.+Weight* model is actually the worst performing model. The difference between the scores is not, however, statistically significant, thus we cannot conclude there is a difference between either model. If this is the case, the *Base Model* would result in the least amount of space required, and additionally it has one of the lowest average query times, making it the most promising candidate simply for those reasons.

### 10.2.4 TREC Washington Post Corpus

Table 10.5 shows the two experiments ran over the TREC Washington Post Corpus, based on the topics and relevance judgments from TREC 2018 Common Core track. The first experiment represents the fallback to classical stage, where we relied on classical link analysis to produce query-independent features. These were then combined with the text-based ranking function through the reranking approaches proposed by Craswell et al. [170], using the best parameters according to their experiments — in particular, we relied on the *sigm* function, with $w = 1.8$, $k = 1$ and $a = 0.6$, to rerank BM25 scores using each of the tested graph-based features. Our goal was to study the impact of Fatigued PageRank, a metric that we proposed based on the idea of fatigued random walks, using the indegree to compute an approximation to the probability of fatigue. We compared it with other link analysis metrics, including indegree, HITS authority, and PageRank. As we can see from the table, all metrics, except indegree, had very little impact in retrieval effectiveness when compared to the BM25 baseline, except for the indegree which had a more visible behavior, decreasing MAP from 0.2031 to 0.1994, but increasing P@10 from ~0.37 to 0.38. The results we obtained on this front were not particularly compelling to keep pursuing, since this is an idea that has been thoroughly explored before, but also because there was no visible gain, particularly based on Fatigued PageRank, the metric that we proposed.

The second experiment shown in Table 10.5 is again based on the hypergraph-of-entity. We indexed the TREC Washington Post Corpus, relying on a text-only version of the representation model (i.e., considering only *term* nodes and *document* hyperedges), as well as on a base model that relied on NER to identify entities and on external knowledge from DBpedia to obtain triples that contained each identified entity. We submitted two runs to TREC 2018 Common Core track, based on the two described hypergraph-of-entity models. Despite the decrease of relevant documents introduced by design in TREC 2018 Common Core track, both of our models severely underperformed, achieving MAP scores of only 0.0070 and 0.0051 and similarly low P@10 scores. Indexing TREC Washington Post Corpus as combined data could not be done directly, like in the case of INEX 2009 Wikipedia collection, since

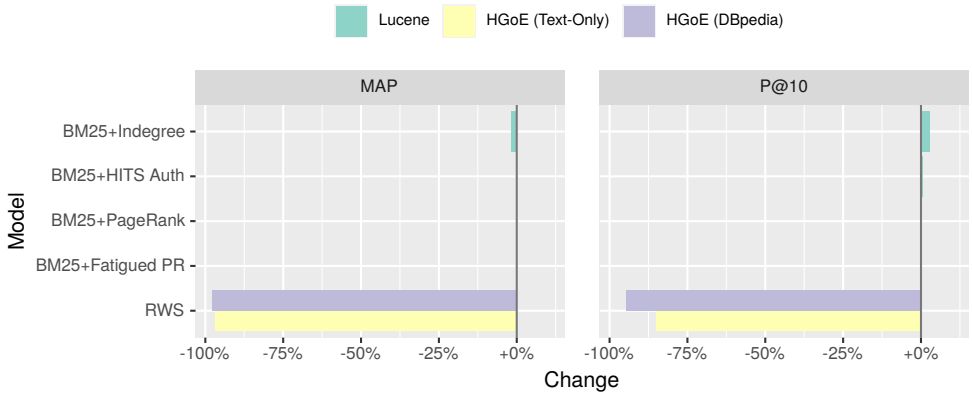



**Figure 10.4:** Relative change in MAP and P@10 compared to the BM25 baseline for each subset of experiments over TREC Washington Post Corpus.

the first was largely approached as an unstructured collection, while the second natively provides semi-structured data. Moreover, we only relied on the first three paragraphs of each document, in order to be able to index the complete collection, which represented an additional challenge. One idea we might have explored if efficiency was not an issue would have been the creation of paragraph hyperedges, since this information was available. Nevertheless, as it stands, results were poor, despite the increased challenge of the 2018 occurrence for the Common Core track.

Figure 10.4 illustrates the change in MAP and P@10 for the two experiments, using BM25 as the baseline. In particular, for the first experiment, we used BM25 scores that we computed, resulting in a MAP of 0.2031 and a P@10 of 0.3700, and, for the second experiment, we used Anserini's BM25 [357], as submitted to TREC 2018 Common Core track with *run_id "anserini_bm25"*, with a MAP of 0.2284 and a P@10 of 0.4500. As we can see, for the first experiment, there is little change, with indegree decreasing MAP, but increasing P@10. Such increase in P@10 and disagreement in metrics is not completely unexpected. For example, in Section B.3.3 we had found that the indegree for Wikipedia's link graph would better approximate user click behavior than PageRank or Fatigued PageRank (cf. Figure B.3). On the other hand, although in that experiment HITS authority would perform better than the indegree for predicting user clicks, the indegree exhibited an inconsistent behavior as the cut size increased. This might explain why P@10 was lower for HITS authority than the indegree.

### 10.2.5 INEX 2009 Wikipedia collection

Table 10.6 illustrates a set of three experiments where, for the first time, and to our knowledge, the viability of a general retrieval model was assessed. In particular, we focused on unifying three tasks from entity-oriented search: (1) ad hoc document retrieval (leveraging entities), (2) ad hoc entity retrieval, and (3) entity list completion. Moreover, in the context of the hypergraph-of-entity, related entity finding can be considered a specialization of entity list completion, achievable by limiting the query to a single entity, making the model actually applicable to four tasks, despite the lack of assessment for the final task.

A fair evaluation of a general retrieval model requires a dataset, usually semi-structured data, that can be interpreted as, or processed to become, combined data (e.g., entity-annotated text, where entities are linked to a knowledge base). Additionally, we must have a set of topics and relevance judgments for the tasks that we want to support in our general retrieval model. This was the case for INEX 2009 Wikipedia collection, which was the only test collection of combined data accompanied by assessments for multiple entity-oriented search tasks. This is why

**Table 10.6:** Overall comparison of retrieval performance, for multiple entity-oriented search tasks, based on the complete INEX 2009 Wikipedia collection.

| Index | Ranking | MAP | P@10 | Index Time | Avg./Query | Nodes | Edges |
|---|---|---|---|---|---|---|---|
| AD HOC DOCUMENT RETRIEVAL | | | | | | | |
| Lucene | TF-IDF | 0.0228 | 0.0692 | 15h 06m | 460ms | – | – |
| | BM25 | 0.0324 | 0.1173 | | 370ms | – | – |
| Fixed parameters over HGoE variations: RWS($\ell = 2, r = 10^4, \Delta_{nf} = 0, \Delta_{ef} = 0, exp = F, wei = F$) | | | | | | | |
| Base Model | RWS | 0.0863 | 0.2462 | 33h 53m | 23s 405ms | 3,506,823 | 7,721,743 |
| Syns | RWS | 0.0937 | 0.2615 | 33h 05m | 55s 555ms | 3,510,462 | 7,734,931 |
| Cont. | RWS | 0.0869 | 0.2654 | 34h 37m | 24s 348ms | 3,604,185 | 7,929,841 |
| TF-Bins$_2$ | RWS | 0.0172 | 0.0500 | 35h 26m | 2m 58s | 3,506,823 | 10,338,867 |
| Syns+Cont. | RWS | 0.0882 | 0.2692 | 37h 16m | 23s 265ms | 3,606,693 | 7,945,083 |
| AD HOC ENTITY RETRIEVAL | | | | | | | |
| Lucene | TF-IDF | 0.0373 | 0.0636 | 59h 17m | 1s 370ms | – | – |
| | BM25 | 0.0668 | 0.1182 | | 798ms | – | – |
| Fixed parameters over HGoE variations: RWS($\ell = 2, r = 10^4, \Delta_{nf} = 0, \Delta_{ef} = 0, exp = F, wei = F$) | | | | | | | |
| Base Model | RWS | 0.1390 | 0.2455 | 33h 53m | 26s 330ms | 3,506,823 | 7,721,743 |
| Syns | RWS | 0.1337 | 0.2473 | 33h 05m | 30s 232ms | 3,510,462 | 7,734,931 |
| Cont. | RWS | 0.1304 | 0.2364 | 34h 37m | 27s 620ms | 3,604,185 | 7,929,841 |
| TF-Bins$_2$ | RWS | 0.0300 | 0.1145 | 35h 26m | 4m 41s | 3,506,823 | 10,338,867 |
| Syns+Cont. | RWS | 0.1313 | 0.2509 | 37h 16m | 26s 877ms | 3,606,693 | 7,945,083 |
| ENTITY LIST COMPLETION | | | | | | | |
| Lucene | TF-IDF | 0.0558 | 0.1000 | 59h 17m | 1s 230ms | – | – |
| | BM25 | 0.0666 | 0.1250 | | 1s 221ms | – | – |
| Fixed parameters over HGoE variations: RWS($\ell = 2, r = 10^4, \Delta_{nf} = 0, \Delta_{ef} = 0, exp = F, wei = F$) | | | | | | | |
| Base Model | RWS | 0.0879 | 0.0769 | 33h 53m | 19s 162ms | 3,506,823 | 7,721,743 |
| Syns | RWS | 0.0857 | 0.0635 | 33h 05m | 19s 875ms | 3,510,462 | 7,734,931 |
| Cont. | RWS | 0.0875 | 0.0692 | 34h 37m | 19s 422ms | 3,604,185 | 7,929,841 |
| TF-Bins$_2$ | RWS | 0.0006 | 0.0058 | 35h 26m | 1m 08s | 3,506,823 | 10,338,867 |
| Syns+Cont. | RWS | 0.0884 | 0.0788 | 37h 16m | 19s 824ms | 3,606,693 | 7,945,083 |

the majority of carried experiments were based on ad hoc document retrieval, leaving this final evaluation of generalization to be run over the INEX 2009 Wikipedia collection. Moreover, due to the scalability issues of our model, and to be able to position our results in regard to the INEX participants from each of the tasks, we opted to use document profiles based on the top 5% keywords. This enabled the index construction over the whole collection to be run in a timely manner, without increasing memory requirements beyond our resources — we had 32 GiB of RAM available, but still needed to create a SWAP file to accommodate the index for the 2.6 million documents, mentioned entities and relations.

As we can see in Table 10.6, we explored three different strategies based on Lucene to provide baselines for each task. For preprocessing text, we simply relied on a `StandardAnalyzer`. For ad hoc document retrieval, we indexed the text block and ran queries using TF-IDF and BM25 with default parameters. For ad hoc entity retrieval, and entity list completion, we relied on entity profiles built from sentences mentioning the entity. We then either used a keyword query to retrieve and rank entities, or an entity query to build a concatenated entity profile that we used as query to retrieve similar entities. This required two indexes, one for the text documents and another one for entity profiles.

For the hypergraph-of-entity indexes, we relied on the preprocessing pipeline described in Section 7.4.2, where we aggressively removed stopwords, filtered small terms, optionally replaced mentions to hyperlinks, time, money and numbers by a common identifier per type, and applied stemming. While we initially attempted to memorize whole sentences, so that we could, at a later stage, explore features like

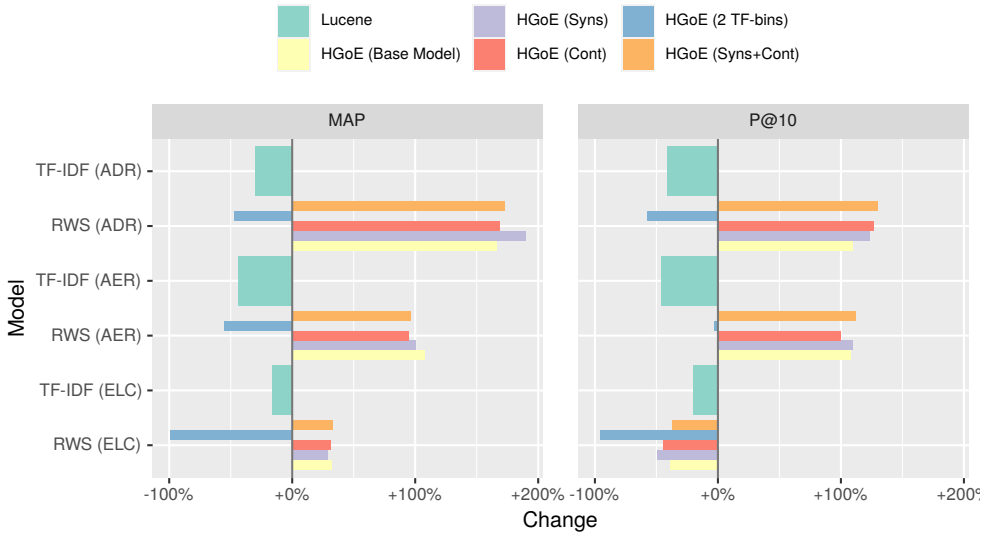



**Figure 10.5:** Relative change in MAP and P@10 compared to the BM25 baseline for each subset of experiments over INEX 2009 Wikipedia collection.

syntactic dependencies, we rapidly discarded this idea, both due to the increased preprocessing time, but also the increased vocabulary space. Thus, we relied on an aggressive analyzer that attempted to reduce the vocabulary to a bare minimum using the well-known classical approaches that we described.

We explored each of the most representative versions of hypergraph-of-entity, with a combination of the two best. We also used the default weight of 0.5 as a kind of probabilistic zero, for all nodes and hyperedges that were not TF-bins, so that the TF-bin weighting would better discriminate high and low TF terms. Overall, and considering the limited information provided by the top 5% of keywords, we were overall able to outperform TF-IDF and BM25 with RWS in most tasks, with the exception of entity list completion, when considering the performance for a cutoff of 10 results (P@10 in the table). The worst overall model was TF-bins$_2$, which unexpectedly performed better when a default weight of zero was used (cf. Table 8.5 in Section 8.5). The best overall model was the hypergraph-of-entity with synonyms for nouns from WordNet 3.0. The task with the best overall MAP scores was ad hoc entity retrieval, although its GMAP (not shown in this table) was much lower, positioning this task at the same level as the remaining two tasks.

For ad hoc document retrieval, BM25 achieved a MAP of 0.03, while our best model, *Syns*, achieved a MAP of 0.0937. For ad hoc entity retrieval, BM25 achieved a MAP of 0.0668, while our best model, *Base Model*, achieved a MAP of 0.1390. Finally, for entity list completion, BM25 achieved a MAP of 0.0666, while our best model, *Syns+Cont.*, achieved a MAP of 0.0884. Although an unfair comparison, due to the limited information provided by the keyword-based document profile, our models still performed quite below the best AP scores for this test collection, which were of 0.4294 (MAP) for the INEX 2010 Ad Hoc track, from participant *"p22-Emse301R"*, of 0.517 (xinfAP) for the entity ranking task of INEX 2009 Entity Ranking track, for run *"2_UAmsISLA_ER_TC_ERreltop"*, and of 0.520 (xinfAP) for the list completion task of INEX 2009 Entity track, for run *"5_UAmsISLA_LC_TE_LCexpTCP"*.

The hypergraph-of-entity remains to be evaluated in the future, for the whole INEX 2009 collection, without relying on keyword-based document profiles for ensuring efficiency. Nevertheless, current results are promising, not only showing that a generalized retrieval model can be built, but also showing that it can, for particular cases, outperform baselines like TF-IDF or even BM25, which still stands as a strong baseline even in 2020.

We close this section by analyzing Figure 10.5, which shows a positive change in MAP and P@10, achieved with the hypergraph-of-entity model. Most versions of our model were able to outperform BM25, improving as much as 190% in MAP for ad hoc document retrieval, or as little as 33% for entity list completion. Another interesting behavior that is visible in the figure is that the overall highest change happened for ad hoc document retrieval, followed by ad hoc entity retrieval, and then entity list completion, at the bottom. This might be explained by the fact that we focused on the first task in most of our experiments, perhaps inadvertently optimizing the universal ranking function and the selected representation model for this task. Additionally, since INEX 2009 Wikipedia collection represents Wikipedia articles, that in turn represent entities, optimizing for these documents might have also resulted in a slight optimization for ad hoc entity retrieval. Future work on the hypergraph-of-entity should perhaps begin with entity list completion for a chance of equally optimizing each task.

## 10.3 LIMITATIONS OF THE HYPERGRAPH-OF-ENTITY

Besides a limited implementation of the core principles, discussed in the previous section, the hypergraph-of-entity has other limitations that we discuss in this section. Namely, we discuss how document preprocessing impacts the indexing process [§10.3.1], the nondeterminism and efficiency issues of the random walk score [§10.3.3], and the pros and cons of using a hypergraph as the central data structure for a general retrieval model [§10.3.4].

### 10.3.1 Document preprocessing

In this thesis, we report indexing times based on the total time required to read and preprocess the collection, adding its terms, entities and relations to the hypergraph-of-entity, and serializing the data structure from RAM to disk. In some cases, however, the reported indexing time is significantly superior than the time required to actually add all terms, entities and relations to the hypergraph. For example, in Section 9.3.4, we reported the indexing time of the hypergraph-of-entity to range between 33h and 37h. In particular, the *Base Model* required, in total, 33h53m13s to be created, however, only 6h24m43s were spent actually indexing the documents. The remaining time was spent in the preprocessing pipeline (not to be confused with the text analysis process required for indexing). This included reading the collection in stream mode directly from the `.tar.bz2` files[1], parsing the XML for each document, to extract text, entities and links, and instantiating a Python *Document* object that was then converted to its Java version through JPype[2]. Only then were documents actually indexed. This means that there is room for optimization simply by means of engineering the code.

### 10.3.2 Keyword extraction for document profiling

For the final experiments (Section 9.3), where we tested the generality of the model, we relied on keyword extraction in order to reduce the size of the index by using document profiles [369]. In particular, we used the TextRank algorithm [363] to identify a ranking of keywords. We then selected the top keywords based on a cutoff ratio (e.g., top 5%). An alternative that we did not test was to instead rely on a constant cutoff value applied over the TextRank scores. Additionally, there are other different approaches for keyword extraction that we might have considered. However, perhaps more importantly, we would like to have run experiments over

1 All code is available as part of Army ANT. Please refer to Section 5.2 for more information.
2 https://jpype.readthedocs.io/en/latest/

an index that contained the complete INEX 2009 Wikipedia collection without the need for reducing each document to its keywords. Alternatively, it would have been useful to at least study performance over increasing cutoff ratios or values, so that we could better predict the expected performance if the full text in the documents were used instead. A related questions is whether a common parameterization for building the keyword-based document profile would result in different performance for different collections, or whether performance would remain stable across collections, in relation to a baseline.

### 10.3.3 Random walk score

At its core, the random walk score is similar to a personalized PageRank, however random walks in hypergraphs are taken in two steps, one to randomly select a hyperedges and another one to randomly select a node from that hyperedge. Due to the nature of our data structure, however, we could not take advantage of the available algebraic mechanisms for simulation (e.g., power iteration). While there is a Multilinear PageRank that is applicable to tensors, that we could experiment with, there is no tensor representation for hypergraphs that supports general mixed hypergraphs and ensures that eigenvectors remain useful. While CERN and the University of Geneva have been working on such a tensor representation [115, 201], their application is for general undirected hypergraphs. With a few progress, this might be an interesting solution for running the random walk score without simulating each step, greatly increasing the performance, while providing access to GPUs for an even greater increase in processing power.

Another issue with the random walk score that we have not completely solved is how to deal with its nondeterministic nature. While it will eventually converge, we are currently relying on the r parameter, which dictates the number of random walks launched from each seed node (either the query term nodes or the adjacent entity nodes). An algebraic implementation based on Multilinear PageRank might inherently solve the convergence problem, rendering this issue irrelevant. However, until this is possible, we might investigate different approaches to ensure that the random walk score only stops iterating after convergence. For now, we propose that the nondeterminism of RWS be dealt with through a long-term caching mechanism to ensure that different users of the search engine have access to a common ranking of results for the same query.

### 10.3.4 Hypergraphs as general structures for information retrieval

During the course of this work, we were faced with challenge for representing term frequencies in our model. The hypergraph was proposed by its author, Claude Berge, as a data structure that "can be used to simplify as well as to generalize". Our initial impression was in fact that this was a flexible data structure, capable of supporting the representation requirements that we incrementally added, as we experimented with the model. However, when confronted with the idea of representing term frequencies, there was no obvious approach. We considered fuzzy hypergraphs, where hyperedges are described by a set of node and weight pairs, which could be used to store term frequencies. The downside of relying on this extension of the hypergraph structure is that it requires the storage of as many weights as the number of edges in the equivalent bipartite graph. We also carried some secondary experiments with the bipartite version of one of our hypergraphs, only to find extremely degraded performance that would not enable us to carry on experiments. Given the likeliness of a fuzzy hypergraph with this bipartite graph, we would say that the challenge of building such a representation and retrieval would be even harder. This is, however, a matter that remains to be considered. Is the hypergraph the ultimate general data structure? Can we go back to graphs and improve performance even more? Should we explore fuzzy hypergraphs [213, 214], or even metagraphs [220]?

## SUMMARY

This chapter was divided into two logical parts, one focused on the experiments prepared throughout this doctoral work, and another on discussing the limitations and possible applications of the proposed model. First, we provided an overview of the carried experiments, looking back on the development process of experimentation, which we divided into five stages: conception, representation model, retrieval model, fallback to classical, and generalization testing. We also analyzed the performance of our models by grouping experiments per test collection, which provided a general view of the impact of each approach. We then covered model limitations, regarding both technical and conceptual issues, including the delay introduced by the document preprocessing stage when compared to the atual indexing time, the impact of keyword extraction in experimentation, the lack of algebraic representation for the random walk score and its efficiency issues, and the hypergraph as an indexing data structure to support general models in information retrieval.

# 11 CONCLUSION

## Contents



Graph-based entity-oriented search is an area that combines the unification proficiency of the graph data structure with the diverse tasks defined to express information needs based on different units of information. Whether users require keywords or entities to articulate their question, and independently of whether they are seeking documents or entities to answer the question, entity-oriented search provides a clear framework that encompasses these diverse tasks. Furthermore, graphs are a data structure capable of representing text, as well as entities, while being used to support a wide range of tasks from multiple domains. Network science is perhaps the most recent area, since cognitive science, that touches on such a wide range of domains.

Appropriately, we relied on graphs, and eventually considered hypergraphs, as tools for developing unified frameworks, in particular with the potential of building generalized models in information retrieval. Motivated by creativity and innovation, we proposed the graph-of-entity, which then evolved into the hypergraph-of-entity, a model that is, to our knowledge, the only graph-based general approach to entity-oriented search.

This thesis has pursued a model for jointly representing corpora and knowledge bases, which we have achieved with the hypergraph-of-entity. It has also pursued the proposal of a universal ranking function, within a general model for information retrieval, which we have achieved with the random walk score, as a way to implement representation-driven retrieval. This work is far from complete. In fact, it raises more questions than it solves. More importantly, it opens a new path of research and attempts to yet again diversify the area of information retrieval, which, after the deep learning absorption, might turn their interest into network science, to find new ways for solving information needs.

The structure of this chapter is organized as follows:

- **Section 11.1** presents a summary of the doctoral work described in this thesis.
- **Section 11.2** concludes this endeavor with some final remarks, summarizing the overall contributions and their impact.
- **Section 11.3** provides several directions for the future, regarding the hypergraph–of-entity and general models for information retrieval.

## 11.1 THESIS SUMMARY

We began this work by providing an historical perspective on information retrieval, and graph and hypergraph theory, link analysis, linked data and graph-based models. Then, we surveyed entity-oriented search contributions that were a result of these combining areas. We presented the motivation for consolidating models, illustrating it with applications in other domains, like physics or machine learning, motivating the graph as the ideal data structure to support such tasks. We then instantiated this problem for the area of entity-oriented search, defining combined data and identifying the retrieval tasks that we would tackle during our attempt at building a generalized retrieval model.

Our state of the art survey required an organization that was non trivial, since our work consisted of proposing a concept that was not yet a part of this community, and it required the integration of knowledge from multiple domains, particularly graph and hypergraph theory, unified models in information retrieval, random walk based approaches, and test collections useful to assess a universal ranking function over a common representation, requiring different topics and relevance judgements for each of the supported tasks.

We relied on empirical research based on test collections, which is common practice in information retrieval, but we also devised a systematic approach for documenting our work, based on a wiki system. In fact, at some point, we were required to repeat multiple experiments, which we could only do thanks to the quality of our documentation system. We also contributed with two large software applications. With ANT, we provided an entity-oriented search engine prototype for the University of Porto, which has been running since 2015, serving the community and reaching over 700 users per week since 2018, a number that grew to 1,600 weekly users following the COVID-19 quarantine. We also developed Army ANT, a workbench for innovation in entity-oriented search and a platform to support the research of general models in information retrieval. This code is open source and it is freely available at GitHub, also providing several Docker images for an accessible installation of the system.

We carried multiple graph-based experiments in the domain of information retrieval, as well as some in the domain of network science, in order to study node ranking approaches and to develop novel hypergraph-based metrics to characterize the data structure. We proposed two main retrieval models, graph-of-entity and hypergraph-of-entity, the former representing the conceptual stage of the latter. We described hypergraph-of-entity in detail, along with the random walk score ranking function, characterizing the hypergraph, while proposing novel approaches for the the analysis of this data structure. We carried several experiments, studying the impact of diverse index extensions applied to the representation model, as well as several parameterizations of the random walk score. We focused on first optimizing for ad hoc document retrieval, and then we assessed the quality of the ad hoc entity retrieval and entity list completion tasks.

In the end, we looked back at our work, documenting the structure of the experimentation approach that we followed, and discussing the overall results per test collection. We presented a reflection on the pillar concepts of information retrieval and how their presence or absence impacted the hypergraph-of-entity. Finally, we discussed the limitations of our model and, further along in this chapter, we propose future applications to extend and reinforce its generality.

## 11.2 FINAL REMARKS

Whether it is possible to successfully generalize information retrieval tasks is perhaps one of the most important questions that we opened with this thesis. Our

proposal for a representation and retrieval model could be considered the first case study for generalization in information retrieval, standing on three tasks from entity-oriented search and relying on graph-based models.

We evaluated several aspects of the hypergraph-of-entity, providing basic statistics about the hypergraph, studying rank stability for a nondeterministic ranking function, and identifying the index extensions and parameter configurations that achieved the best performance. In some cases, we were able to obtain MAP scores comparable to Lucene TF-IDF, or even surpass BM25 when relying on keyword-based document profiles. More importantly, we were able to capture and integrate heterogeneous information from corpora and knowledge bases in a joint representation model that provides explicit semantics and extreme flexibility in the definition of $n$-ary relations, such as synonyms and context, as well as the subsumption or hierarchical relations. We showed that the hypergraph-of-entity is significantly more efficient than the graph-of-entity in indexing time and that it can also be configured, through parameter $r$, for faster search times albeit with a penalty in effectiveness. One of the more evident limitations of the model is the lack of consideration for document frequency and document length. Although verbosity is mitigated (term repetitions are not considered), vocabulary diversity in long documents that cover multiple topics is still a problem (a kind of pivoted document length normalization is required). Despite its performance limitations, particularly when compared to state-of-the-art approaches, hypergraph-based representations have the potential to more naturally model our cognition process, unlocking increasingly intelligent information retrieval systems as we study and approach the brain.

Our main goal was to propose and evaluate a graph-based general model for entity-oriented search. Our focus was on whether we would be able to adequately support three specific tasks: ad hoc document retrieval (leveraging entities), ad hoc entity retrieval, and entity list completion. We compared the effectiveness of each task over the hypergraph-of-entity, with the lowest ranking runs of participants in the INEX 2009 and 2010 respective tasks. We found an overall lower performance, that consistently scaled with state-of-the-art evaluation scores (i.e., tasks that had generally higher scores, when compared with the remaining tasks, also had higher scores in our model). However, for all the tested tasks, and according to MAP, GMAP, NDCG@10 and P@10, we were able to outperform Lucene TF-IDF and BM25 when representing documents by their keyword-based profile, with the exception of the entity list completion task based on P@10 and NDCG@10. This showed the potential for an effective hypergraph-of-entity model capable of supporting retrieval generalization. Despite its low overall performance, we have demonstrated that a unified framework for entity-oriented search can be built, and we have opened several new opportunities for contributions in improving the performance of the hypergraph-of-entity, motivating the proposal of new hypergraph-based retrieval models, or even the exploration of novel general retrieval models, while studying the advantages of this new approach.

## 11.3 FUTURE WORK

In this section, we present several ideas for the future, both at a high-level of abstraction, and at a high-level of detail. We cover the representation model [§11.3.1], the characterization of hypergraphs [§11.3.2], the random walk score [§11.3.3], and even promote the generalization of the hypergraph-of-entity model based on new, unexplored applications [§11.3.4].

### 11.3.1 Representation model

There are several pending improvements over the hypergraph-of-entity, even regarding basic tasks, like exploring the ideal preprocessing pipeline. More importantly, future work should focus on alternative approaches to reduce the complexity of the representation model, such as the document profiles based on keywords that we proposed here. Can we automatically identify nodes or hyperedges that can be removed? How would other keyword extraction algorithms work, or how would other ratio values or even a fixed cutoff value perform to select the top keywords? We would have liked to measure the impact of pruning nodes and hyperedges from the hypergraph, independently for each type of node and hyperedge, as well as based on different weight thresholds. Another alternative that we would have liked to explore was based on pruning similar hyperedges (e.g., based on subhypergraph similarities, which is in itself a complex task). The goal is to understand how much we can improve efficiency until effectiveness begins to degrade. In the same line of work, we would also like to explore alternative approaches for generating the context similarity network that we relied upon, using different word embedding strategies, as well as avoiding non-optimal algorithms, like k-nearest neighbors, to obtain the top similar terms. An alternative would easily be the usage of pivots for an approximated measurement of similarity [370].

Apart from reducing the model by pruning redundancies, we would, on the other hand, like to extend it with synonyms for verbs, adjectives and adverbs, measuring the impact in effectiveness, and understanding whether the usage of *synsets* for nouns had been sufficient. Another interesting idea, that has been shown to improve query understanding [293], is the usage of dependency parsing. It would be interesting to extract term dependencies from the documents in a collection, building a dependency graph and integrating those relations into the hypergraph (like we did for the word2vec similarity network). The idea is that it might indirectly improve query understanding, even for simple keyword queries, and thus positively impact the overall retrieval effectiveness. On the other hand, efficiency must be significantly improved before being able to run such experiments at a medium/large scale. Still regarding representation, we would like to better justify the usage of *entity* nodes as opposed to *entity* hyperedges, which is the approach proposed by Bendersky and Croft in their query hypergraph [14]. We would also like to revise *related_to* hyperedges in order to better harness the information provided by triples associated with a document, exploring other options beyond grouping by subject.

While we focused on improving the efficiency of graph-of-entity by defining a new hypergraph-of-entity model, there are still scalability issues to be tackled. In particular, we would like to asses how the model scales over different datasets, for example the DBLP co-authorship network. We predict that, as the size of the collection increases, efficiency problems will become more prominent and we think this might also be mitigated by using different approaches to compute random walks, namely based on fingerprinting, as described by Fogaras et al. [237] or Chakrabarti [67].

Despite having identified the capacity for an hypergraph-based model to capture subsumption and hierarchical relations, we have not assessed the impact of such relations in retrieval effectiveness. This is something that could be explored in the future. Regarding node and hyperedge weighting functions, there are still many opportunities to explore. We are interested in studying node and hyperedge weighting functions, with the goal of finding uniformly distributed functions with a high discriminative power. We have previously explored biased random walks with basic weighting functions, but they were skewly distributed, which was actually harmful to overall performance. Finding such weighting functions could also provide a way to prune the hypergraph. Finally, it is also not clear whether weights can be learned automatically, perhaps through micro machine learning models positioned in nodes and hyperedges, or whether such weighting models should be dependent on the

target domain or query intent. Furthermore, there are several approaches for tackling *tf_bin* hyperedge weighting by deriving a value based on the term frequencies of the hyperedge members.

### 11.3.2 Hypergraph characterization

Regarding hypergraph characterization, we would like to further explore the computation of density, since the bipartite-graph-based density that we proposed, although useful, only accounts for hyperedges already in the hypergraph. We would also like to study the parameterization of the two estimation approaches that we proposed for computing shortest distances and clustering coefficients, based on random walks and node sampling, respectively. Another open challenge is the definition of a random hypergraph generation model, which would be useful to improve characterization, as it would provide a way to compare the clustering coefficient with that of an equivalent random hypergraph, perhaps identifying the existence of a small-world equivalent for hypergraphs. Several opportunities also exist in the study of the hypergraph at a mesoscale, be it identifying communities, network motifs or graphlets, or exploring unique patterns to hypergraphs. Additionally, it would be interesting to include centrality metrics in the correlation analysis, in order to understand for instance whether closeness or betweenness might impact retrieval effectiveness in the hypergraph-of-entity, while considering multiple combinations of index extensions.

### 11.3.3 Random walk score

Random walks in a hypergraph [55] can be seen as a form of randomized sampling of the structure of the hypergraph. The longer the random walk or the higher the number of repeats, the better the ability to capture hypergraph structure. This means that, assuming random walks do their job of correctly sampling structure, the representation model will then be the fundamental indicator of retrieval effectiveness, hence representation-driven retrieval.

The current ranking approach is based on simulating individual steps of the random walk, but ideally this would be based on a Markov process over a matrix or tensor representation of the general mixed hypergraph that is the foundation for our model. We could then take advantage of the GPU for improving efficiency. Nevertheless, there are several challenges in this front. Only recently has CERN been studying algebraic approaches for representing general hypergraphs, using adjacency tensors [115, 201]. However, our hypergraph is mixed (i.e., containing both directed and undirected hyperedges), which has not yet been explored. Furthermore, given the recency of this work, there are still no widespread tools for working with these tensors.

Regarding the random walk score, which is nondeterministic, could we find a parameter configuration, or a different approach, that always leads to convergence, no matter the dataset? Or could we instead experiment with reranking algorithms, such as learning to rank approaches? Would caching be sufficient? Could our model compete with multi-task learning to provide rankings for the three different tasks?

We should also further experiment with IDF, beyond the probabilistic IDF proposed as the weighting function for term nodes in Section 7.4.1.4, instead modifying the hypergraph-of-entity so that it structurally provides an analogy to IDF instead of explicitly relying on a precomputed value that will require biased random walks, which have been shown to delay retrieval. Furthermore, we should also consider mitigating the issue with high cardinality *document* hyperedges, in analogy to the issue with long documents, particular with a diverse vocabulary, that is classically solved by pivoted document length normalization. One way to introduce a better

analogy to the behavior of pivoted document length normalization could be, for instance, the replacement of traversals through *document* hyperedges by traversals through *sentence* hyperedges, therefore providing a natural method for normalization by restructuring the representation model.

Hypergraph structure acts as a constraint for random walking — while random walks in the Euclidean space can essentially take a step in any direction, in hypergraphs they are restricted to taking steps within the structure of the hypergraph. The idea of fatigue that we explore here is simply an added restriction, similar to the one we introduce when moving from the Euclidean space to a hypergraph space. Regarding fatigued random walks, it would be interesting to reiterate over the computation approach of Fatigued PageRank, exploring a more approximate analogy to the combination of PageRank and Reverse PageRank, replacing the $k^*$ vector with the Markov matrix used in Reverse PageRank and studying the differences. Assuming we could represent hypergraph-of-entity as a tensor, we could also attempt to introduce the concept of fatigue in the Multilinear PageRank, so that we could completely rebuild the random walk score in its algebraic version, solvable through power iteration or other available, more efficient methods of PageRank computation, such as Monte Carlo (see Section A.2).

### 11.3.4 Promoting generalization through new applications

We proposed what is, to our knowledge, the first general model for information retrieval. While we explored multiple tasks from entity-oriented search, some of them analogous to recommendation, there are many other information retrieval and information filtering tasks, that might be solvable through our model, either with slight or no changes at all. In this section, we briefly present some of these ideas, to motivate the continued research of a model that truly solves general information needs. We classify these applications into user-based [§11.3.4.1], query-based [§11.3.4.2] and text-based [§11.3.4.3] tasks.

#### 11.3.4.1 *User-based tasks*

Adding user nodes to the hypergraph would enable us to create hyperedges modeling previous query terms, possibly in bins, like with did for TF-bins, so that we could distinguish between terms and entities (i.e., the topics) that are of a particular interest to a each user.

**PERSONALIZED SEARCH** A user seed node or hyperedge, representing an authenticated user, could be a part of the search process, where we could even control the degree of personalization by boosting the weight of the user node and adjacent hyperedges, or the nodes in the user hyperedge, depending on the chosen modeling approach.

**EXPERT SEARCH** In analogy to expert search, users could search for people, either based on similar interests, or with a search profile that could promote trust of that individual as a potential expert at a given topic.

#### 11.3.4.2 *Query-based tasks*

While user nodes and search profiles would work as a grouping of searched keywords, we might explicitly define query hyperedges, storing the terms most frequently co-occurring in queries.

**QUERY SUGGESTION** Query autocompletion could be supported on user profiles, query hyperedges, or even the terms from the already indexed documents. Without the cost of building an external data structure and a specialized approach for query

suggestion, we might promptly rely on the hypergraph for this task, since it already contains the required information.

**QUERY ENTITY LINKING** In the hypergraph-of-entity model that we propose in this thesis, there is already a *contained_in* hyperedge that establishes relations between terms and entities, however, there is much to be done until a robust query entity linking system can be implemented over the hypergraph. In particular, we have not explored the links among the entities connected to a set of terms, which are so frequently useful for disambiguation tasks. This remains to be explored.

#### 11.3.4.3 *Text-based tasks*

There are also some text-based tasks that we envisioned might be possible to implement with the hypergraph-of-entity. Some of them, however, relied on the idea that the hypergraph would retain all terms along with the syntactic structure of the sentences. This is an idea that was not viable to explore, due to the severe impact in efficiency that the addition of *sentence* hyperedges had, as well as due to the increment in preprocessing time. However, a less structured version of such ideas might still provide interesting results.

**TEXT SUMMARIZATION** One of these ideas was generative summarization. With sentences and their syntactic structure intact, we might be able to generate new sentences from a given document or set of documents, in order to provide a summary of a given subject or topic. Another possibility would be extractive summarization, again due to the fact that the model would retain syntactic structure. However, in its current implementation, in order for efficient search to be pursued, the solution we adopted was to greatly reduce the size of the vocabulary. With this version we might still be able to do keyword or topic extraction, based on small chains of terms used in queries or small documents.

**TEXT AUGMENTATION** We describe text augmentation as the general task of illustrating text through related external elements, such as entities. The idea is inspired by text illustration work carried by researchers like Filipe Coelho [370], who relied on image search using a full document or passage as a query.

**ENTITY DESCRIPTION** Given a particular entity, retrieve the documents that best describe the context of this entity. This task should be straightforward to implement in the current version of hypergraph-of-entity, however an evaluation framework would be required to assess the quality of the results.

**LEXICON CONSTRUCTION** Given the abundance of terms, linked to entities and their relations, it would be interesting to provide an automated lexicon construction approach that could support information science experts in the task of building domain-specific lexicons, through the indexing of relevant collections or through the search of a relevant topic, perhaps implemented as a subhypergraph ranking task.

## SUMMARY

The challenges in joint representation models and universal ranking functions are immense. They pave the way for unified information retrieval, where a general retrieval model might support tasks like search and recommendation, but also subtasks like query expansion, and word sense or named entity disambiguation. In this thesis, we have explored multiple different ideas, that we have integrated in the domain of entity-oriented search, through graph-based models, to propose the first general retrieval model over a joint representation model of corpora and knowledge bases. The hypergraph-of-entity can solve multiple tasks based on a single model and a universal ranking function. We have also paved the way with an immense list of ideas for future work around the hypergraph-of-entity, hypergraph theory and analysis, and the random walk score, also proposing several different applications to further extend and generalize the hypergraph-of-entity to new tasks. It is our sincere hope that this thesis will inspire a different way of thinking about information retrieval, opening up the area to new scientific talent and pushing it forward in a different way.

# BIBLIOGRAPHY


[1] S. Gurajada, J. Kamps, A. Mishra, R. Schenkel, M. Theobald, and Q. Wang. "Overview of the INEX 2013 Linked Data Track". In: *Working Notes for CLEF 2013 Conference , Valencia, Spain, September 23-26, 2013*. 2013. URL: http://ceur-ws.org/Vol-1179/CLEF2013wn-INEX-GurajadaEt2013.pdf (cit. on p. xv).

[2] T. Berners-Lee, J. Hendler, O. Lassila, et al. "The semantic web". In: *Scientific american* 284.5 (2001), pp. 28–37. URL: https://www.jstor.org/stable/26059207 (cit. on pp. 1, 8, 48).

[3] J. Pound, P. Mika, and H. Zaragoza. "Ad-hoc object retrieval in the web of data". In: *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*. Ed. by M. Rappa, P. Jones, J. Freire, and S. Chakrabarti. ACM, 2010, pp. 771–780. DOI: 10.1145/1772690.1772769 (cit. on pp. 1, 11, 33, 71, 93, 103, 116, 119, 331).

[4] M. Bautin and S. Skiena. "Concordance-Based Entity-Oriented Search". In: *2007 IEEE / WIC / ACM International Conference on Web Intelligence, WI 2007, 2-5 November 2007, Silicon Valley, CA, USA, Main Conference Proceedings*. IEEE Computer Society, 2007, pp. 586–592. DOI: 10.1109/WI.2007.84 (cit. on pp. 1, 10, 12, 31, 33, 36, 71, 128, 162, 230, 331).

[5] A. Singhal. *Official Google Blog: Introducing the Knowledge Graph: things, not strings*. https://googleblog.blogspot.pt/2012/05/introducing-knowledge-graph-things-not.html. Accessed on 2017-04-11. May 2012 (cit. on pp. 2, 9, 334).

[6] M. L. Spruyt. "The place of cataloguing and classification in the curricula of South African universities". PhD thesis. University of Cape Town, 1980 (cit. on p. 3).

[7] D. Harman. "Information Retrieval: The Early Years". In: *Foundations and Trends in Information Retrieval* 13.5 (2019), pp. 425–577. DOI: 10.1561/1500000065 (cit. on p. 3).

[8] H. P. Luhn. "A Statistical Approach to Mechanized Encoding and Searching of Literary Information". In: *IBM Journal of Research and Development* 1.4 (1957), pp. 309–317. DOI: 10.1147/rd.14.0309 (cit. on pp. 3, 5, 14, 31, 75, 226, 227).

[9] K. S. Jones. "A statistical interpretation of term specificity and its application in retrieval". In: *J. Documentation* 60.5 (2004 [1972]), pp. 493–502. DOI: 10.1108/00220410410560573 (cit. on pp. 4, 5, 31, 226, 227, 234).

[10] A. Singhal, G. Salton, M. Mitra, and C. Buckley. "Document Length Normalization". In: *Inf. Process. Manage.* 32.5 (1996), pp. 619–633. DOI: 10.1016/0306-4573(96)00008-8 (cit. on pp. 4, 6, 31).

[11] A. Singhal, C. Buckley, and M. Mitra. "Pivoted Document Length Normalization". In: *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'96, August 18-22, 1996, Zurich, Switzerland (Special Issue of the SIGIR Forum)*. 1996, pp. 21–29. DOI: 10.1145/243199.243206 (cit. on pp. 4, 6, 31, 232, 234).

[12] D. Hiemstra. "Information Retrieval Models". In: *Information Retrieval*. John Wiley & Sons, Ltd, 2009. Chap. 1, pp. 1–19. ISBN: 9780470033647. DOI: 10.1002/9780470033647.ch1 (cit. on pp. 4, 31).

[13] J. Zobel and A. Moffat. "Inverted files for text search engines". In: *ACM Comput. Surv.* 38.2 (2006), p. 6. DOI: 10.1145/1132956.1132959 (cit. on p. 4).

[14] M. Bendersky and W. B. Croft. "Modeling Higher-order Term Dependencies in Information Retrieval Using Query Hypergraphs". In: *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '12. Portland, Oregon, USA: ACM, 2012, pp. 941–950. ISBN: 978-1-4503-1472-5. DOI: 10.1145/2348283.2348408 (cit. on pp. 4, 7, 14, 54, 71, 72, 163, 257, 335).

[15] R. Blanco and C. Lioma. "Graph-based term weighting for information retrieval". In: *Information Retrieval* 15.1 (2012), pp. 54–92. DOI: 10.1007/s10791-011-9172-x (cit. on pp. 4, 7, 9, 13, 44, 45, 71, 72, 149, 151, 152, 162, 169, 328, 333).

[16] F. Rousseau and M. Vazirgiannis. "Graph-of-word and TW-IDF: new approach to ad hoc IR". In: *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013*. Ed. by Q. He, A. Iyengar, W. Nejdl, J. Pei, and R. Rastogi. ACM, 2013, pp. 59–68. DOI: 10.1145/2505515.2505671 (cit. on pp. 4, 7, 9, 13, 22, 32, 45, 55, 70, 71, 83, 84, 129, 149, 151, 152, 154, 158, 160, 162, 169, 220, 233, 239, 333).

[17] G. Boole. *The mathematical analysis of logic*. Philosophical Library, 1847 (cit. on p. 4).

[18] V. Bush. "As We May Think". In: *The Atlantic Monthly* 176.1 (1945), pp. 101–108. URL: http://www.theatlantic.com/unbound/flashbks/computer/bushf.htm (cit. on p. 4).

[19] M. E. Maron and J. L. Kuhns. "On Relevance, Probabilistic Indexing and Information Retrieval". In: *J. ACM* 7.3 (1960), pp. 216–244. DOI: 10.1145/321033.321035 (cit. on p. 5).

[20] N. Jardine and C. J. van Rijsbergen. "The use of hierarchic clustering in information retrieval". In: *Information Storage and Retrieval* 7.5 (1971), pp. 217–240. DOI: 10.1016/0020-0271(71)90051-9 (cit. on p. 5).

[21] E. M. Voorhees. "The Cluster Hypothesis Revisited". In: *Proceedings of the 8th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Montréal, Québec, Canada, June 5-7, 1985*. 1985, pp. 188–196. DOI: 10.1145/253495.253524 (cit. on p. 5).

[22] D. E. Knuth. *The Art of Computer Programming, Volume III: Sorting and Searching*. Addison-Wesley, 1973. ISBN: 0-201-03803-X (cit. on p. 5).

[23] A. Bookstein and D. R. Swanson. "Probabilistic models for automatic indexing". In: *JASIS* 25.5 (1974), pp. 312–316. DOI: 10.1002/asi.4630250505 (cit. on p. 5).

[24] S. P. Harter. "A probabilistic approach to automatic keyword indexing. Part II. An algorithm for probabilistic indexing". In: *JASIS* 26.5 (1975), pp. 280–289. DOI: 10.1002/asi.4630260504 (cit. on pp. 5, 32).

[25] S. E. Robertson and K. S. Jones. "Relevance weighting of search terms". In: *JASIS* 27.3 (1976), pp. 129–146. DOI: 10.1002/asi.4630270302 (cit. on p. 6).

[26] C. Faloutsos and S. Christodoulakis. "Signature Files: An Access Method for Documents and Its Analytical Performance Evaluation". In: *ACM Trans. Inf. Syst.* 2.4 (1984), pp. 267–288. DOI: 10.1145/2275.357411 (cit. on pp. 6, 73).

[27] N. Fuhr. "Optimum Polynomial Retrieval Functions Based on the Probability Ranking Principle". In: *ACM Trans. Inf. Syst.* 7.3 (1989), pp. 183–204. DOI: 10.1145/65943.65944 (cit. on p. 6).

[28] U. Manber and G. Myers. "Suffix Arrays: A New Method for On-Line String Searches". In: *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms, 22-24 January 1990, San Francisco, California, USA*. Ed. by D. S. Johnson. SIAM, 1990, pp. 319–327. URL: http://dl.acm.org/citation.cfm?id=320176.320218 (cit. on p. 6).

[29] H. R. Turtle and W. B. Croft. "Evaluation of an Inference Network-Based Retrieval Model". In: *ACM Trans. Inf. Syst.* 9.3 (1991), pp. 187–222. DOI: 10.1145/125187.125188 (cit. on pp. 6, 32).

[30] F. J. Burkowski. "Retrieval Activities in a Database Consisting of Heterogeneous Collections of Structured Text". In: *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. Copenhagen, Denmark, June 21-24, 1992*. 1992, pp. 112–125. DOI: 10.1145/133160.133185 (cit. on p. 6).

[31] S. E. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. "Okapi at TREC-3". In: *Proceedings of The Third Text REtrieval Conference, TREC 1994, Gaithersburg, Maryland, USA, November 2-4, 1994*. Ed. by D. K. Harman. Vol. 500-225. NIST Special Publication. National Institute of Standards and Technology (NIST), 1994, pp. 109–126. URL: http://trec.nist.gov/pubs/trec3/papers/city.ps.gz (cit. on pp. 6, 32, 233).

[32] D. Bouchaffra and J. G. Meunier. "A Markovian random field approach to information retrieval". In: *Third International Conference on Document Analysis and Recognition, ICDAR 1995, August 14 - 15, 1995, Montreal, Canada. Volume II*. IEEE Computer Society, 1995, pp. 997–1002. DOI: 10.1109/ICDAR.1995.602070 (cit. on p. 6).

[33] L. Page. *PageRank: Bringing order to the web*. Tech. rep. Stanford Digital Libraries Working Paper, Sept. 1997. URL: http://www-diglib.stanford.edu/diglib/WP/PUBLIC/DOC159.html (cit. on pp. 6, 8, 56, 299, 312).

[34] J. M. Kleinberg. *Autoritative Sources in a Hyperlinked Environment*. Tech. rep. RJ 10076 (91892). Almaden Research Center, 650 Harry Road, San Jose California 95120-6099: IBM Research Division, May 1997. URL: http://www.cs.cornell.edu/home/kleinber/auth.pdf (cit. on pp. 6, 8).

[35] J. M. Ponte and W. B. Croft. "A Language Modeling Approach to Information Retrieval". In: *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 24-28 1998, Melbourne, Australia*. 1998, pp. 275–281. DOI: 10.1145/290941.291008 (cit. on pp. 6, 32, 72).

[36] I. Ounis, G. Amati, V. Plachouras, B. He, C. Macdonald, and C. Lioma. "Terrier: A High Performance and Scalable Information Retrieval Platform". In: *Proceedings of ACM SIGIR'06 Workshop on Open Source Information Retrieval (OSIR 2006)*. Seattle, Washington, USA, 2006. URL: http://terrierteam.dcs.gla.ac.uk/publications/ounis06terrier-osir.pdf (cit. on pp. 7, 122).

[37] O. Jespersen. *The philosophy of grammar*. Routledge, 2013 [1924]. URL: 10.4324/9780203716045 (cit. on pp. 7, 44).

[38] Q. Ai, X. Wang, S. Bruch, N. Golbandi, M. Bendersky, and M. Najork. "Learning Groupwise Multivariate Scoring Functions Using Deep Neural Networks". In: *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval, ICTIR 2019, Santa Clara, CA, USA, October 2-5, 2019*. 2019, pp. 85–92. DOI: 10.1145/3341981.3344218 (cit. on pp. 7, 38, 129).

[39] A. Bavelas. "Communication Patterns in Task-Oriented Groups". In: *The Journal of the Acoustical Society of America* 22.6 (Nov. 1950), pp. 725–730. DOI: 10.1121/1.1906679 (cit. on pp. 8, 43).

[40] C. Berge. *Graphes et hypergraphes*. Dunod: Paris, 1970 (cit. on pp. 8, 14, 53, 165, 186, 188).

[41] P. Bonacich. "Technique for Analyzing Overlapping Memberships". In: *Sociological Methodology* 4 (1972), p. 176. DOI: 10.2307/270732 (cit. on p. 8).

[42] P. Bonacich. "Factoring and weighting approaches to status scores and clique identification". In: *The Journal of Mathematical Sociology* 2.1 (1972), pp. 113–120. DOI: 10.1080/0022250X.1972.9989806 (cit. on p. 8).

[43] F. Göbel and A. Jagers. "Random walks on graphs". In: *Stochastic processes and their applications* 2.4 (1974), pp. 311–336. DOI: 10.1016/0304-4149(74)90001-5 (cit. on p. 8).

[44] L. C. Freeman. "A Set of Measures of Centrality Based on Betweenness". In: *Sociometry* 40.1 (Mar. 1977), p. 35. DOI: 10.2307/3033543 (cit. on pp. 8, 43).

[45] T. J. Berners-Lee. *Information management: a proposal*. Tech. rep. CERN-DD-89-001-OC. March 1989 version includes the annotations from Mike Sendall, Tim Berners-Lee's supervisor. Geneva: CERN, Mar. 1989. URL: https://cds.cern.ch/record/369245 (cit. on p. 8).

[46] R. Cyganiak, D. Wood, and M. Lanthaler. "RDF 1.1 concepts and abstract syntax". In: (2014). URL: https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/ (cit. on p. 8).

[47] B. Adida, M. Birbeck, S. McCarron, and S. Pemberton. "RDFa in XHTML: Syntax and Processing — A collection of attributes and processing rules for extending XHTML to support RDF". In: *World Wide Web Consortium, Recommendation REC-rdfa-syntax-20081014* (2008). URL: http://www.w3.org/TR/2008/REC-rdfa-syntax-20081014 (cit. on pp. 9, 169).

[48] R. Khare and T. Çelik. "Microformats: a pragmatic path to the semantic web". In: *Proceedings of the 15th International Conference on World Wide Web, WWW 2006, Edinburgh, Scotland, UK, May 23-26, 2006*. 2006, pp. 865–866. DOI: 10.1145/1135777.1135917 (cit. on p. 9).

[49] K. D. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. "Freebase: a collaboratively created graph database for structuring human knowledge". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*. 2008, pp. 1247–1250. DOI: 10.1145/1376616.1376746 (cit. on pp. 9, 13).

[50] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. G. Ives. "DBpedia: A Nucleus for a Web of Open Data". In: *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007*. 2007, pp. 722–735. DOI: 10.1007/978-3-540-76298-0_52 (cit. on pp. 9, 13, 48, 55, 162, 222).

[51] M. Lanthaler and C. Gütl. "On using JSON-LD to create evolvable RESTful services". In: *Third International Workshop on RESTful Design, WS-REST '12, Lyon, France, April 16, 2012*. 2012, pp. 25–32. DOI: 10.1145/2307819.2307827 (cit. on p. 9).

[52] Bing, Google, and Yahoo! *Schema.org*. https://schema.org/. June 2011 (cit. on p. 9).

[53] D. Vrandecic. "Wikidata: a new platform for collaborative data collection". In: *Proceedings of the 21st World Wide Web Conference, WWW 2012, Lyon, France, April 16-20, 2012 (Companion Volume)*. 2012, pp. 1063–1064. DOI: 10.1145/2187980.2188242 (cit. on pp. 9, 13).

[54] R. Qian. *Bing Blogs: Understand Your World with Bing*. https://blogs.bing.com/search/2013/03/21/understand-your-world-with-bing/. Accessed 2019-05-27. Mar. 2013 (cit. on pp. 9, 71, 334).

[55] A. Bellaachia and M. Al-Dhelaan. "Random walks in hypergraph". In: *Proceedings of the 2013 International Conference on Applied Mathematics and Computational Methods, Venice Italy*. 2013, pp. 187–194. URL: http://www.inase.org/library/2013/venice/bypaper/AMCM/AMCM-28.pdf (cit. on pp. 9, 55, 69, 258, 335).

[56] D. Herrmannova and P. Knoth. "An Analysis of the Microsoft Academic Graph". In: *D-Lib Mag.* 22.9/10 (2016). DOI: 10.1045/september2016-herrmannova (cit. on p. 9).

[57] J. Zhong, H. Zhu, J. Li, and Y. Yu. "Conceptual Graph Matching for Semantic Search". In: *Conceptual Structures: Integration and Interfaces, 10th International Conference on Conceptual Structures, ICCS 2002, Borovets, Bulgaria, July 15-19, 2002, Proceedings*. Ed. by U. Priss, D. Corbett, and G. Angelova. Vol. 2393. Lecture Notes in Computer Science. Springer, 2002, pp. 92–196. DOI: 10.1007/3-540-45483-7_8 (cit. on pp. 9, 51, 71, 334).

[58] H. Zhu, J. Zhong, J. Li, and Y. Yu. "An Approach for Semantic Search by Matching RDF Graphs". In: *Proceedings of the Fifteenth International Florida Artificial Intelligence Research Society Conference, May 14-16, 2002, Pensacola Beach, Florida, USA*. Ed. by S. M. Haller and G. Simmons. AAAI Press, 2002, pp. 450–454. URL: http://www.aaai.org/Library/FLAIRS/2002/flairs02-088.php (cit. on pp. 9, 51, 71, 334).

[59] N. Stojanovic and L. Stojanovic. "Searching for the Knowledge in the Semantic Web". In: *Proceedings of the Fifteenth International Florida Artificial Intelligence Research Society Conference, May 14-16, 2002, Pensacola Beach, Florida, USA*. 2002, pp. 435–439. URL: http://www.aaai.org/Library/FLAIRS/2002/flairs02-085.php (cit. on p. 9).

[60] R. V. Guha, R. McCool, and E. Miller. "Semantic search". In: *Proceedings of the Twelfth International World Wide Web Conference, WWW 2003, Budapest, Hungary, May 20-24, 2003*. 2003, pp. 700–709. DOI: 10.1145/775152.775250 (cit. on p. 9).

[61] D. Bonino, F. Corno, L. Farinetti, and A. Bosca. "Ontology driven semantic search". In: *WSEAS Transaction on Information Science and Application* 1.6 (2004), pp. 1597–1605. URL: http://www.wseas.us/e-library/conferences/venice2004/papers/472-334.pdf (cit. on pp. 9, 10).

[62] K. Balog. *Entity-Oriented Search*. Vol. 39. The Information Retrieval Series. Springer, 2018. ISBN: 978-3-319-93933-9. DOI: 10.1007/978-3-319-93935-3 (cit. on pp. 9, 13, 16, 29, 31, 33, 36, 41, 56, 68, 72, 125, 299, 331).

[63] S. Pepper. *The TAO of topic maps: Finding the Way in the Age of Infoglut*. Tech. rep. STEP Infotek, 2000. URL: https://ontopia.net/topicmaps/materials/tao.pdf (cit. on p. 10).

[64] S. R. Newcomb, N. A. Kipp, and V. T. Newcomb. "The "HyTime" Hypermedia/Time-based Document Structuring Language". In: *Commun. ACM* 34.11 (1991), pp. 67–83. DOI: 10.1145/125490.125495 (cit. on p. 10).

[65] A. Balmin, V. Hristidis, and Y. Papakonstantinou. "ObjectRank: Authority-Based Keyword Search in Databases". In: *(e)Proceedings of the Thirtieth International Conference on Very Large Data Bases, Toronto, Canada, August 31 - September 3 2004*. 2004, pp. 564–575. URL: http://www.vldb.org/conf/2004/RS15P2.PDF (cit. on pp. 10, 57, 59, 71, 299, 335).

[66] A. Hogan, A. Harth, and S. Decker. "ReConRank: A Scalable Ranking Method for Semantic Web Data with Context". In: *Proceedings of Second International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2006), in conjunction with International Semantic Web Conference (ISWC 2006)*. 2006. URL: http://hdl.handle.net/10379/492 (cit. on pp. 10, 56, 59, 71, 299, 335).

[67] S. Chakrabarti. "Dynamic personalized PageRank in entity-relation graphs". In: *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*. 2007, pp. 571–580. DOI: 10.1145/1242572.1242650 (cit. on pp. 10, 58, 59, 71, 257, 299, 335).

[68] A. P. de Vries, A. Vercoustre, J. A. Thom, N. Craswell, and M. Lalmas. "Overview of the INEX 2007 Entity Ranking Track". In: *Focused Access to XML Documents, 6th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2007, Dagstuhl Castle, Germany, December 17-19, 2007. Selected Papers*. 2007, pp. 245–251. DOI: 10.1007/978-3-540-85902-4_22 (cit. on pp. 10, 67, 88).

[69] K. Balog, P. Serdyukov, and A. P. de Vries. "Overview of the TREC 2010 Entity Track". In: *Proceedings of The Nineteenth Text REtrieval Conference, TREC 2010, Gaithersburg, Maryland, USA, November 16-19, 2010*. Ed. by E. M. Voorhees and L. P. Buckland. Vol. 500-294. NIST Special Publication. National Institute of Standards and Technology (NIST), 2010. URL: http://trec.nist.gov/pubs/trec19/papers/ENTITY.OVERVIEW.pdf (cit. on pp. 10, 60, 67, 88).

[70] K. Balog, A. P. de Vries, P. Serdyukov, P. Thomas, and T. Westerveld. "Overview of the TREC 2009 Entity Track". In: *Proceedings of The Eighteenth Text REtrieval Conference, TREC 2009, Gaithersburg, Maryland, USA, November 17-20, 2009*. 2009. URL: http://trec.nist.gov/pubs/trec18/papers/ENT09.OVERVIEW.pdf (cit. on pp. 10, 67, 88).

[71] Y. Wu and H. Kashioka. "NiCT at TREC 2009: Employing Three Models for Entity Ranking Track". In: *Proceedings of The Eighteenth Text REtrieval Conference, TREC 2009, Gaithersburg, Maryland, USA, November 17-20, 2009*. 2009. URL: http://trec.nist.gov/pubs/trec18/papers/nict.ENT.pdf (cit. on p. 10).

[72] R. Delbru, N. Toupikov, M. Catasta, G. Tummarello, and S. Decker. "Hierarchical Link Analysis for Ranking Web Data". In: *The Semantic Web: Research and Applications, 7th Extended Semantic Web Conference, ESWC 2010, Heraklion, Crete, Greece, May 30 - June 3, 2010, Proceedings, Part II*. 2010, pp. 225–239. DOI: 10.1007/978-3-642-13489-0_16 (cit. on pp. 10, 58, 59, 71, 72, 299, 335).

[73] H. Raviv, O. Kurland, and D. Carmel. "The cluster hypothesis for entity oriented search". In: *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information etrieval (SIGIR 2013)*. 2013, p. 841. ISBN: 9781450320344. DOI: 10.1145/2484028.2484128 (cit. on pp. 11, 35, 44, 71, 72, 331).

[74] H. Bast and B. Buchhold. "An Index for Efficient Semantic Full-text Search". In: *Proceedings of the 22Nd ACM International Conference on Conference on Information and Knowledge Management*. 2013, pp. 369–378. ISBN: 978-1-4503-2263-8. DOI: 10.1145/2505515.2505689 (cit. on pp. 11, 17, 35, 71, 162, 331).

[75] N. Zhiltsov and E. Agichtein. "Improving entity search over linked data by modeling latent semantics". In: *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013*. 2013, pp. 1253–1256. DOI: 10.1145/2505515.2507868 (cit. on pp. 11, 50, 71, 334).

[76] R. Haentjens Dekker and D. J. Birnbaum. "It's more than just overlap: Text As Graph". In: *Proceedings of Balisage: The Markup Conference 2017*. Vol. 19. 2017. DOI: 10.4242/BalisageVol19.Dekker01 (cit. on pp. 11, 14, 55, 71, 76, 163, 335).

[77] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008. ISBN: 978-0-521-86571-5. DOI: 10.1017/CBO9780511809071. URL: https://nlp.stanford.edu/IR-book/pdf/irbookprint.pdf (cit. on pp. 11, 13, 310).

[78] A. Einstein. "Erklärung der Perihelionbewegung der Merkur aus der allgemeinen Relativitätstheorie". In: *Sitzungsberichte der Königlich Preußischen Akademie der Wissenschaften (Berlin)* 47.2 (Jan. 1915), pp. 831–839 (cit. on p. 12).

[79] J. von Neumann. *The Computer and the Brain (The Silliman Memorial Lectures Series)*. New Haven, CT: Yale University Press, 2012 (cit. on pp. 12, 23, 314, 315).

[80] A. Newell. *Unified Theories of Cognition*. Harvard University Press, Jan. 1994. ISBN: 978-0-674-92101-6 (cit. on p. 12).

[81] J. R. Anderson and C. Lebiere. "The Newell test for a theory of cognition". In: *Behavioral and Brain Sciences* 26.5 (2003), pp. 587–601. DOI: 10.1017/S0140525X0300013X (cit. on p. 12).

[82] X. S. Zhou and T. S. Huang. "Unifying Keywords and Visual Contents in Image Retrieval". In: *IEEE MultiMedia* 9.2 (2002), pp. 23–33. DOI: 10.1109/93.998050 (cit. on p. 12).

[83] J. Bu, S. Tan, C. Chen, C. Wang, H. Wu, L. Zhang, and X. He. "Music recommendation by unified hypergraph: combining social media information and music content". In: *Proceedings of the 18th International Conference on Multimedia 2010, Firenze, Italy, October 25-29, 2010*. 2010, pp. 391–400. DOI: 10.1145/1873951.1874005 (cit. on pp. 12, 55, 335).

[84] A. Moro, A. Raganato, and R. Navigli. "Entity Linking meets Word Sense Disambiguation: a Unified Approach". In: *Trans. Assoc. Comput. Linguistics* 2 (2014), pp. 231–244. URL: https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/291 (cit. on pp. 12, 14, 15, 162, 181).

[85] P. Domingos. *The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World*. Penguin Books, 2015. ISBN: 978-0-141-97924-3 (cit. on pp. 12, 149).

[86] C. V. Gysel, M. de Rijke, and E. Kanoulas. "Learning Latent Vector Spaces for Product Search". In: *Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*. 2016, pp. 165–174. DOI: 10.1145/2983323.2983702 (cit. on pp. 12, 40, 71, 162, 332).

[87] N. J. Belkin and W. B. Croft. "Information Filtering and Information Retrieval: Two Sides of the Same Coin?" In: *Commun. ACM* 35.12 (1992), pp. 29–38. DOI: 10.1145/138859.138861 (cit. on pp. 12, 72, 164).

[88] H. Zamani and W. B. Croft. "Joint Modeling and Optimization of Search and Recommendation". In: *Proceedings of the First Biennial Conference on Design of Experimental Search & Information Retrieval Systems, Bertinoro, Italy, August 28-31, 2018*. 2018, pp. 36–41. URL: http://ceur-ws.org/Vol-2167/paper2.pdf (cit. on p. 12).

[89] A. Emtage and P. Deutsch. "Archie: An electronic directory service for the internet". In: *Proceedings of the USENIX Winter 1992 Technical Conference*. San Francisco, CA, USA, 1992, pp. 93–110 (cit. on p. 12).

[90] M. Arrington. *AOL Proudly Releases Massive Amounts of Private Data*. https://techcrunch.com/2006/08/06/aol-proudly-releases-massive-amounts-of-user-search-data/. Accessed on 2017-07-13. Aug. 2006 (cit. on p. 12).

[91] E. F. T. K. Sang and F. D. Meulder. "Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition". In: *Proceedings of the Seventh Conference on Natural Language Learning, CoNLL 2003, Held in cooperation with HLT-NAACL 2003, Edmonton, Canada, May 31 - June 1, 2003*. 2003, pp. 142–147. URL: http://aclweb.org/anthology/W/W03/W03-0419.pdf (cit. on p. 12).

[92] H. Bast, B. Buchhold, and E. Haussmann. "Semantic Search on Text and Knowledge Bases". In: *Found. Trends Inf. Retr.* 10.2-3 (2016), pp. 119–271. DOI: 10.1561/1500000032 (cit. on pp. 13, 16, 48, 61, 62, 76, 121, 149, 150, 162, 185).

[93] R. Baeza-Yates, M. Ciaramita, P. Mika, and H. Zaragoza. "Towards Semantic Search". In: *Natural Language and Information Systems, 13th International Conference on Applications of Natural Language to Information Systems, NLDB 2008, London, UK, June 24-27, 2008, Proceedings*. Ed. by E. Kapetanios, V. Sugumaran, and M. Spiliopoulou. Vol. 5039. Lecture Notes in Computer Science. Springer, 2008, pp. 4–11. DOI: 10.1007/978-3-540-69858-6_2 (cit. on p. 13).

[94] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer. "DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia". In: *Semantic Web* 6.2 (2015), pp. 167–195. DOI: 10.3233/SW-140134 (cit. on pp. 13, 124).

[95] F. M. Suchanek, G. Kasneci, and G. Weikum. "YAGO: a core of semantic knowledge". In: *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*. 2007, pp. 697–706. DOI: 10.1145/1242572.1242667 (cit. on pp. 13, 48, 72).

[96] B. Fields, C. Rhodes, M. A. Casey, and K. Jacobson. "Social Playlists and Bottleneck Measurements: Exploiting Musician Social Graphs Using Content-Based Dissimilarity and Pairwise Maximum Flow Values". In: *ISMIR 2008, 9th International Conference on Music Information Retrieval, Drexel University, Philadelphia, PA, USA, September 14-18, 2008*. 2008, pp. 559–564. URL: http://ismir2008.ismir.net/papers/ISMIR2008_209.pdf (cit. on p. 13).

[97] Y. Jing and S. Baluja. "VisualRank: Applying PageRank to Large-Scale Image Search". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 30.11 (2008), pp. 1877–1890. DOI: 10.1109/TPAMI.2008.121 (cit. on pp. 13, 305, 312).

[98] E. Yan and Y. Ding. "Applying centrality measures to impact analysis: A coauthorship network analysis". In: *JASIST* 60.10 (2009), pp. 2107–2118. DOI: 10.1002/asi.21128 (cit. on p. 13).

[99] O. Sporns. *Networks of the Brain*. Cambridge, MA, USA: MIT Press, 2010. ISBN: 978-0-262-01469-4 (cit. on p. 13).

[100] J.-F. Rual, K. Venkatesan, T. Hao, T. Hirozane-Kishikawa, A. Dricot, N. Li, G. F. Berriz, F. D. Gibbons, M. Dreze, N. Ayivi-Guedehoussou, et al. "Towards a proteome-scale map of the human protein–protein interaction network". In: *Nature* 437.7062 (Sept. 2005), pp. 1173–1178. DOI: 10.1038/nature04209 (cit. on p. 13).

[101] P. Mika. "Social Networks and the Semantic Web". PhD thesis. Vrije Universiteit Amsterdam, Dec. 2006 (cit. on p. 14).

[102] S. Koschade. "A Social Network Analysis of Jemaah Islamiyah: The Applications to Counterterrorism and Intelligence". In: *Studies in Conflict & Terrorism* 29.6 (2006), pp. 559–575. DOI: 10.1080/10576100600798418 (cit. on p. 14).

[103] L. Dietz. "ENT Rank: Retrieving Entities for Topical Information Needs through Entity-Neighbor-Text Relations". In: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*. 2019, pp. 215–224. DOI: 10.1145/3331184.3331257 (cit. on pp. 14, 55, 71, 335).

[104] M. Richardson and P. M. Domingos. "Markov logic networks". In: *Machine Learning* 62.1-2 (2006), pp. 107–136. DOI: 10.1007/s10994-006-5833-1 (cit. on p. 14).

[105] R. Biagioni, P. Vandenbussche, and V. Novácek. "Finding Explanations of Entity Relatedness in Graphs: A Survey". In: *CoRR* abs/1809.07685 (2018). arXiv: 1809.07685. URL: http://arxiv.org/abs/1809.07685 (cit. on pp. 14, 73).

[106] M. Bendersky. "Information retrieval with query hypergraphs". In: *SIGIR Forum* 46.2 (2012), p. 111. DOI: 10.1145/2422256.2422273 (cit. on p. 14).

[107] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. "Open Information Extraction from the Web". In: *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*. Ed. by M. M. Veloso. 2007, pp. 2670–2676. URL: http://ijcai.org/Proceedings/07/Papers/429.pdf (cit. on pp. 17, 108).

[108] A. Fader, S. Soderland, and O. Etzioni. "Identifying Relations for Open Information Extraction". In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*. 2011, pp. 1535–1545. URL: https://www.aclweb.org/anthology/D11-1142/ (cit. on pp. 17, 18).

[109] J. Zhu, D. Song, and S. M. Rüger. "Integrating Document Features for Entity Ranking". In: *Focused Access to XML Documents, 6th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2007, Dagstuhl Castle, Germany, December 17-19, 2007. Selected Papers*. 2007, pp. 336–347. DOI: 10.1007/978-3-540-85902-4_29 (cit. on p. 20).

[110] R. Reinanda, E. Meij, J. Pantony, and J. J. Dorando. "Related Entity Finding on Highly-heterogeneous Knowledge Graphs". In: *IEEE/ACM 2018 International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2018, Barcelona, Spain, August 28-31, 2018*. 2018, pp. 330–334. DOI: 10.1109/ASONAM.2018.8508650 (cit. on pp. 20, 332).

[111] F. Rousseau and M. Vazirgiannis. "Main Core Retention on Graph-of-Words for Single-Document Keyword Extraction". In: *Advances in Information Retrieval - 37th European Conference on IR Research, ECIR 2015, Vienna, Austria,*

*March 29 - April 2, 2015. Proceedings*. Ed. by A. Hanbury, G. Kazai, A. Rauber, and N. Fuhr. Vol. 9022. Lecture Notes in Computer Science. 2015, pp. 382–393. DOI: 10.1007/978-3-319-16354-3_42 (cit. on pp. 22, 74, 226, 227).

[112] R. Schenkel, F. M. Suchanek, and G. Kasneci. "YAWN: A Semantically Annotated Wikipedia XML Corpus". In: *Datenbanksysteme in Business, Technologie und Web (BTW 2007), 12. Fachtagung des GI-Fachbereichs "Datenbanken und Informationssysteme" (DBIS), Proceedings, 7.-9. März 2007, Aachen, Germany*. 2007, pp. 277–291. URL: http://subs.emis.de/LNI/Proceedings/Proceedings103/article1404.html (cit. on pp. 22, 48, 84, 93, 122, 127, 192).

[113] L. Page, S. Brin, R. Motwani, and T. Winograd. *The PageRank Citation Ranking: Bringing Order to the Web.* Technical Report 1999-66. Previous number = SIDL-WP-1999-0120. Stanford InfoLab, Nov. 1999. URL: http://ilpubs.stanford.edu:8090/422/ (cit. on pp. 23, 42, 56, 70, 71, 74, 302, 304, 305, 312, 316, 333).

[114] D. F. Gleich, L. Lim, and Y. Yu. "Multilinear PageRank". In: *SIAM J. Matrix Analysis Applications* 36.4 (2015), pp. 1507–1541. DOI: 10.1137/140985160 (cit. on pp. 23, 42, 56, 72, 299, 310, 313).

[115] X. Ouvrard, J. L. Goff, and S. Marchand-Maillet. "Adjacency and Tensor Representation in General Hypergraphs Part 1: e-adjacency Tensor Uniformisation Using Homogeneous Polynomials". In: *CoRR* abs/1712.08189 (2017). arXiv: 1712.08189 (cit. on pp. 23, 53, 73, 185, 240, 252, 258).

[116] M. Gupta and M. Bendersky. "Information Retrieval with Verbose Queries". In: *Foundations and Trends in Information Retrieval* 9.3-4 (2015), pp. 91–208. DOI: 10.1561/1500000050 (cit. on p. 31).

[117] H. Fang, T. Tao, and C. Zhai. "A formal study of information retrieval heuristics". In: *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Sheffield, UK, July 25-29, 2004*. 2004, pp. 49–56. DOI: 10.1145/1008992.1009004 (cit. on pp. 32, 45, 154).

[118] Y. Lv and C. Zhai. "Lower-bounding term frequency normalization". In: *Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011, Glasgow, United Kingdom, October 24-28, 2011*. 2011, pp. 7–16. DOI: 10.1145/2063576.2063584 (cit. on pp. 32, 45, 152).

[119] G. Amati and C. J. van Rijsbergen. "Probabilistic models of information retrieval based on measuring the divergence from randomness". In: *ACM Transactions on Information Systems* 20.4 (2002), pp. 357–389. DOI: 10.1145/582415.582416 (cit. on p. 32).

[120] D. Metzler and W. B. Croft. "A Markov random field model for term dependencies". In: *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2005)*. 2005, p. 472. ISBN: 1595930345. DOI: 10.1145/1076034.1076115 (cit. on pp. 32, 37, 72).

[121] L. Lloyd, D. Kechagias, and S. Skiena. "Lydia: A System for Large-Scale News Analysis". In: *String Processing and Information Retrieval, 12th International Conference, SPIRE 2005, Buenos Aires, Argentina, November 2-4, 2005, Proceedings*. Ed. by M. P. Consens and G. Navarro. Vol. 3772. Lecture Notes in Computer Science. Springer, 2005, pp. 161–166. DOI: 10.1007/11575832_18 (cit. on p. 33).

[122] R. Fagin, R. Kumar, and D. Sivakumar. "Comparing Top k Lists". In: *SIAM J. Discrete Math.* 17.1 (2003), pp. 134–160. URL: http://epubs.siam.org/sam-bin/dbq/article/41285 (cit. on p. 33).

[123] R. Bhagdev, S. Chapman, F. Ciravegna, V. Lanfranchi, and D. Petrelli. "Hybrid Search: Effectively Combining Keywords and Semantic Searches". In:

*The Semantic Web: Research and Applications, 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008, Proceedings*. Ed. by S. Bechhofer, M. Hauswirth, J. Hoffmann, and M. Koubarakis. Vol. 5021. Lecture Notes in Computer Science. Springer, 2008, pp. 554–568. DOI: 10.1007/978-3-540-68234-9_41 (cit. on pp. 33, 71, 331).

[124] H. Raviv, D. Carmel, and O. Kurland. "A ranking framework for entity oriented search using Markov random fields". In: *Proceedings of the 1st Joint International Workshop on Entity-Oriented and Semantic Search (JIWES 2012)*. 2012, pp. 1–6. ISBN: 9781450316019. DOI: 10.1145/2379307.2379308 (cit. on pp. 33, 34, 36, 71, 72, 331).

[125] C. L. Koumenides and N. R. Shadbolt. "Combining link and content-based information in a Bayesian inference model for entity search". In: *Proceedings of the 1st Joint International Workshop on Entity-Oriented and Semantic Search - JIWES '12*. 2012, pp. 1–6. ISBN: 9781450316019. DOI: 10.1145/2379307.2379310 (cit. on pp. 34, 71, 331).

[126] J. Urbain. "User-Driven Relational Models for Entity-Relation Search and Extraction". In: *Proceedings of the 1st Joint International Workshop on Entity-Oriented and Semantic Search*. JIWES '12. Portland, Oregon, USA: Association for Computing Machinery, 2012. ISBN: 9781450316019. DOI: 10.1145/2379307.2379312 (cit. on pp. 34, 71, 331).

[127] E. Yilmaz and J. A. Aslam. "Estimating average precision with incomplete and imperfect judgments". In: *Proceedings of the 2006 ACM CIKM International Conference on Information and Knowledge Management, Arlington, Virginia, USA, November 6-11, 2006*. 2006, pp. 102–111. DOI: 10.1145/1183614.1183633 (cit. on p. 34).

[128] M. Bron, K. Balog, and M. de Rijke. "Example Based Entity Search in the Web of Data". In: *Advances in Information Retrieval - 35th European Conference on IR Research, ECIR 2013, Moscow, Russia, March 24-27, 2013. Proceedings*. 2013, pp. 392–403. DOI: 10.1007/978-3-642-36973-5_33 (cit. on pp. 35, 48, 71, 331).

[129] H. Bast, F. Bäurle, B. Buchhold, and E. Haussmann. "Broccoli: Semantic Full-Text Search at your Fingertips". In: *CoRR* abs/1207.2615 (2012). arXiv: 1207.2615. URL: http://arxiv.org/abs/1207.2615 (cit. on pp. 35, 162).

[130] M. Zhou. "Entity-Centric Search: Querying by Entities and for Entities". PhD thesis. University of Illinois at Urbana-Champaign, 2014. URL: http://hdl.handle.net/2142/72748 (cit. on pp. 36, 71, 331).

[131] L. Dietz and M. Schuhmacher. "An Interface Sketch for Queripidia: Query-driven Knowledge Portfolios from the Web". In: *Proceedings of the Eighth Workshop on Exploiting Semantic Annotations in Information Retrieval, ESAIR 2015, Melbourne, Australia, October 23, 2015*. Ed. by K. Balog, J. Dalton, A. Doucet, and Y. Ibrahim. ACM, 2015, pp. 43–46. DOI: 10.1145/2810133.2810145 (cit. on pp. 36, 162, 331).

[132] L. Dietz, M. Schuhmacher, and S. P. Ponzetto. "Queripidia: Query-specific wikipedia construction". In: *Proceedings of the 4th Workshop on Automated Knowledge Base Construction (AKBC 2014)* (2014). URL: http://ciir-publications.cs.umass.edu/pub/web/getpdf.php?id=1174 (cit. on pp. 37, 71).

[133] H. Li. "A Short Introduction to Learning to Rank". In: *IEICE Transactions on Information and Systems* E94-D.10 (2011), pp. 1–2. ISSN: 0916-8532. DOI: 10.1587/transinf.E94.D.1 (cit. on p. 37).

[134] T. Liu. *Learning to Rank for Information Retrieval*. Springer, 2011. ISBN: 978-3-642-14266-6. DOI: 10.1007/978-3-642-14267-3 (cit. on p. 37).

[135] C. J. C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. N. Hullender. "Learning to rank using gradient descent". In: *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005*. 2005, pp. 89–96. DOI: 10.1145/1102351.1102363 (cit. on p. 38).

[136] C. J. C. Burges, R. Ragno, and Q. V. Le. "Learning to Rank with Nonsmooth Cost Functions". In: *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*. 2006, pp. 193–200. URL: http://papers.nips.cc/paper/2971-learning-to-rank-with-nonsmooth-cost-functions (cit. on p. 38).

[137] R. Chen, D. Spina, W. B. Croft, M. Sanderson, and F. Scholer. "Harnessing Semantics for Answer Sentence Retrieval". In: *Proceedings of the Eighth Workshop on Exploiting Semantic Annotations in Information Retrieval, ESAIR 2015, Melbourne, Australia, October 23, 2015*. Ed. by K. Balog, J. Dalton, A. Doucet, and Y. Ibrahim. ACM, 2015, pp. 21–27. DOI: 10.1145/2810133.2810136 (cit. on pp. 38, 71, 332).

[138] D. Metzler and T. Kanungo. "Machine Learned Sentence Selection Strategies for Query-Biased Summarization". In: *Proceedings of SIGIR 2008 Workshop on Learning to Rank for Information Retrieval (LR4IR), held in conjunction with the 31th Annual International ACM SIGIR Conference*. Singapore, July 2008, pp. 40–47 (cit. on p. 38).

[139] E. Gabrilovich and S. Markovitch. "Computing Semantic Relatedness Using Wikipedia-based Explicit Semantic Analysis". In: *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*. Ed. by M. M. Veloso. 2007, pp. 1606–1611. URL: http://ijcai.org/Proceedings/07/Papers/259.pdf (cit. on pp. 38, 50).

[140] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. "Distributed Representations of Words and Phrases and their Compositionality". In: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*. 2013, pp. 3111–3119. URL: http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality (cit. on pp. 38, 175).

[141] B. Lin, K. D. Rosa, R. Shah, and N. Agarwal. "LADS: Rapid Development of a Learning-To-Rank Based Related Entity Finding System using Open Advancement". In: *Proceedings of The First International Workshop on Entity-Oriented Search (EOS)*. 2011 (cit. on pp. 38, 71, 332).

[142] K. Balog, P. Serdyukov, and A. P. de Vries. "Overview of the TREC 2011 Entity Track". In: *Proceedings of The Twentieth Text REtrieval Conference, TREC 2011, Gaithersburg, Maryland, USA, November 15-18, 2011*. 2011. URL: http://trec.nist.gov/pubs/trec20/papers/ENTITY.OVERVIEW.pdf (cit. on pp. 38, 63, 88).

[143] M. Schuhmacher, L. Dietz, and S. P. Ponzetto. "Ranking Entities for Web Queries Through Text and Knowledge". In: *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015*. 2015, pp. 1461–1470. DOI: 10.1145/2806416.2806480 (cit. on pp. 39, 71, 332).

[144] J. Chen, C. Xiong, and J. Callan. "An Empirical Study of Learning to Rank for Entity Search". In: *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July 17-21, 2016*. 2016, pp. 737–740. DOI: 10.1145/2911451.2914725 (cit. on pp. 39, 71, 72, 332).

[145] R. Caruana. "Multitask Learning". In: *Machine Learning* 28.1 (1997), pp. 41–75. DOI: 10.1023/A:1007379606734 (cit. on p. 40).

[146] J. Bai, K. Zhou, G. Xue, H. Zha, G. Sun, B. L. Tseng, Z. Zheng, and Y. Chang. "Multi-task learning for learning to rank in web search". In: *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009*. 2009, pp. 1549–1552. DOI: 10.1145/1645953.1646169 (cit. on p. 40).

[147] M. Fernández, I. Cantador, V. López, D. Vallet, P. Castells, and E. Motta. "Semantically enhanced Information Retrieval: An ontology-based approach". In: *J. Web Semant.* 9.4 (2011), pp. 434–452. DOI: 10.1016/j.websem.2010.11.003 (cit. on pp. 41, 148).

[148] L. Getoor and C. P. Diehl. "Link Mining: A Survey". In: *SIGKDD Explor. Newsl.* 7.2 (Dec. 2005), pp. 3–12. ISSN: 1931-0145. DOI: 10.1145/1117454.1117456 (cit. on pp. 41, 300).

[149] J. M. Kleinberg. "Authoritative Sources in a Hyperlinked Environment". In: *J. ACM* 46.5 (1999), pp. 604–632. DOI: 10.1145/324133.324140 (cit. on pp. 42, 56, 71, 333).

[150] M. Saerens and F. Fouss. "HITS is Principal Components Analysis". In: *2005 IEEE / WIC / ACM International Conference on Web Intelligence (WI 2005), 19-22 September 2005, Compiegne, France*. 2005, pp. 782–785. DOI: 10.1109/WI.2005.71 (cit. on p. 42).

[151] S. Brin and L. Page. "The Anatomy of a Large-Scale Hypertextual Web Search Engine". In: *Comput. Networks* 30.1-7 (1998), pp. 107–117. DOI: 10.1016/S0169-7552(98)00110-X (cit. on pp. 42, 56, 71, 303, 312, 333).

[152] D. Gleich, L. Zhukov, and P. Berkhin. *Fast parallel PageRank: A linear system approach*. Tech. rep. YRL-2004-038. Yahoo! Research, 2004. URL: http://research.yahoo.com/publication/YRL-2004-038.pdf (cit. on p. 42).

[153] G. M. D. Corso, A. Gulli, and F. Romani. "Fast PageRank Computation via a Sparse Linear System". In: *Internet Mathematics* 2.3 (2005), pp. 251–273. DOI: 10.1080/15427951.2005.10129108 (cit. on p. 42).

[154] K. Avrachenkov, N. Litvak, D. Nemirovsky, and N. Osipova. "Monte Carlo Methods in PageRank Computation: When One Iteration is Sufficient". In: *SIAM J. Numerical Analysis* 45.2 (2007), pp. 890–904. DOI: 10.1137/050643799 (cit. on pp. 42, 56, 302).

[155] D. F. Gleich. "PageRank Beyond the Web". In: *SIAM Review* 57.3 (2015), pp. 321–363. DOI: 10.1137/140976649 (cit. on pp. 42, 56, 72, 300).

[156] T. H. Haveliwala. "Topic-Sensitive PageRank: A Context-Sensitive Ranking Algorithm for Web Search". In: *IEEE Trans. Knowl. Data Eng.* 15.4 (2003), pp. 784–796. DOI: 10.1109/TKDE.2003.1208999 (cit. on pp. 42, 56, 307, 313).

[157] D. Dimitrov, P. Singer, F. Lemmerich, and M. Strohmaier. "What Makes a Link Successful on Wikipedia?" In: *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*. 2017, pp. 917–926. DOI: 10.1145/3038912.3052613 (cit. on pp. 42, 56, 99, 309, 313, 324, 325).

[158] X. Wang, T. Tao, J. Sun, A. Shakery, and C. Zhai. "DirichletRank: Solving the zero-one gap problem of PageRank". In: *ACM Trans. Inf. Syst.* 26.2 (2008), 10:1–10:29. DOI: 10.1145/1344411.1344416 (cit. on pp. 42, 73, 306, 312).

[159] D. Fogaras. "Where to Start Browsing the Web?" In: *Innovative Internet Community Systems, Third International Workshop, IICS 2003, Leipzig, Germany, June 19-21, 2003, Revised Papers*. 2003, pp. 65–79. DOI: 10.1007/978-3-540-39884-4_6 (cit. on p. 42).

[160] Z. Bar-Yossef and L. Mashiach. "Local approximation of PageRank and reverse PageRank". In: *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2008, Singapore, July 20-24, 2008*. 2008, pp. 865–866. DOI: 10.1145/1390334.1390545 (cit. on p. 42).

[161] Z. Gyöngyi, H. Garcia-Molina, and J. O. Pedersen. "Combating Web Spam with TrustRank". In: *(e)Proceedings of the Thirtieth International Conference on Very Large Data Bases, Toronto, Canada, August 31 - September 3 2004*. 2004, pp. 576–587. URL: http://www.vldb.org/conf/2004/RS15P3.PDF (cit. on p. 42).

[162] T. Van and M. Beigbeder. "Web Co-citation: Discovering Relatedness Between Scientific Papers". In: *Advances in Intelligent Web Mastering, Proceedings of the 5th Atlantic Web Intelligence Conference - AWIC 2007, Fontainebleau, France, June 25 - 27, 2007*. 2007, pp. 343–348. DOI: 10.1007/978-3-540-72575-6_55 (cit. on pp. 43, 50, 71, 333).

[163] T. Ito, M. Shimbo, T. Kudo, and Y. Matsumoto. "Application of kernels to link analysis". In: *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, Illinois, USA, August 21-24, 2005*. 2005, pp. 586–592. DOI: 10.1145/1081870.1081941 (cit. on pp. 43, 71, 333).

[164] F. Chung. "The heat kernel as the pagerank of a graph". In: *Proceedings of the National Academy of Sciences* 104.50 (2007), pp. 19735–19740. ISSN: 0027-8424. DOI: 10.1073/pnas.0708838104 (cit. on pp. 43, 52, 71, 300, 305, 333).

[165] K. Kloster and D. F. Gleich. "Heat kernel based community detection". In: *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*. 2014, pp. 1386–1395. DOI: 10.1145/2623330.2623706 (cit. on pp. 43, 44, 52, 71, 333).

[166] F. van Rest. "A mathematical approach to scalable personalized PageRank". Bachelor thesis. Mathematisch Instituut, Universiteit Leiden, May 2009. URL: https://www.math.leidenuniv.nl/scripties/vanRestBach.pdf (cit. on pp. 43, 333).

[167] C. Engström and S. Silvestrov. "A componentwise pagerank algorithm". In: *16th Applied Stochastic Models and Data Analysis International Conference (ASMDA2015) with Demographics 2015 Workshop, 30 June–4 July 2015, University of Piraeus, Greece*. ISAST: International Society for the Advancement of Science and Technology. 2015, pp. 185–198. URL: http://www.asmda.es/images/1_E-G_ASMDA2015_Proceedings.pdf (cit. on p. 43).

[168] R. Andersen, F. R. K. Chung, and K. J. Lang. "Local Graph Partitioning using PageRank Vectors". In: *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*. 2006, pp. 475–486. DOI: 10.1109/FOCS.2006.44 (cit. on pp. 44, 56).

[169] R. Yang, X. Xiao, Z. Wei, S. S. Bhowmick, J. Zhao, and R. Li. "Efficient Estimation of Heat Kernel PageRank for Local Clustering". In: *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*. 2019, pp. 1339–1356. DOI: 10.1145/3299869.3319886 (cit. on pp. 44, 71, 333).

[170] N. Craswell, S. E. Robertson, H. Zaragoza, and M. J. Taylor. "Relevance weighting for query independent evidence". In: *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Salvador, Brazil, August 15-19, 2005*. 2005, pp. 416–423. DOI: 10.1145/1076034.1076106 (cit. on pp. 44, 247, 327).

[171] Í. C. Dourado, R. Galante, M. A. Gonçalves, and R. da Silva Torres. "Bag of textual graphs (BoTG): A general graph-based text representation model". In: *J. Assoc. Inf. Sci. Technol.* 70.8 (2019), pp. 817–829. DOI: 10.1002/asi.24167 (cit. on pp. 45, 71, 333).

[172] M. Fernández, V. López, M. Sabou, V. S. Uren, D. Vallet, E. Motta, and P. Castells. "Semantic Search Meets the Web". In: *Proceedings of the 2th IEEE International Conference on Semantic Computing (ICSC 2008), August 4-7, 2008, Santa Clara, California, USA*. IEEE Computer Society, 2008, pp. 253–260. DOI: 10.1109/ICSC.2008.52 (cit. on pp. 46, 71, 333).

[173] V. López, M. Sabou, and E. Motta. "PowerMap: Mapping the Real Semantic Web on the Fly". In: *The Semantic Web - ISWC 2006, 5th International Semantic Web Conference, ISWC 2006, Athens, GA, USA, November 5-9, 2006, Proceedings*. 2006, pp. 414–427. DOI: 10.1007/11926078_30 (cit. on p. 46).

[174] K. Byrne. "Populating the semantic web — combining text and relational databases as RDF graphs". PhD thesis. Institute for Communicating and Collaborative Systems, School of Informatics, University of Edinburgh, 2009. URL: http://hdl.handle.net/1842/3781 (cit. on pp. 46, 71, 334).

[175] K. Balog, M. de Rijke, R. Franz, H. Peetz, B. Brinkman, I. Johgi, and M. Hirschel. "SaHaRa: Discovering Entity-Topic Associations in Online News". In: *8th International Semantic Web Conference (ISWC 2009)*. 2009 (cit. on pp. 47, 71, 333).

[176] R. Blanco, P. Mika, and S. Vigna. "Effective and Efficient Entity Search in RDF Data". In: *The Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I*. 2011, pp. 83–97. DOI: 10.1007/978-3-642-25073-6_6 (cit. on pp. 47, 71, 334).

[177] J. Herrera, A. Hogan, and T. Käfer. "BTC-2019: The 2019 Billion Triple Challenge Dataset". In: *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part II*. 2019, pp. 163–180. DOI: 10.1007/978-3-030-30796-7_11 (cit. on p. 47).

[178] S. E. Robertson and H. Zaragoza. "The Probabilistic Relevance Framework: BM25 and Beyond". In: *Foundations and Trends in Information Retrieval* 3.4 (2009), pp. 333–389. DOI: 10.1561/1500000019 (cit. on p. 47).

[179] R. Neumayer, K. Balog, and K. Nørvåg. "On the Modeling of Entities for Ad-Hoc Entity Search in the Web of Data". In: *Advances in Information Retrieval - 34th European Conference on IR Research, ECIR 2012, Barcelona, Spain, April 1-5, 2012. Proceedings*. 2012, pp. 133–145. DOI: 10.1007/978-3-642-28997-2_12 (cit. on pp. 47, 48, 71, 173, 334).

[180] A. Tonon, M. Catasta, R. Prokofyev, G. Demartini, K. Aberer, and P. Cudré-Mauroux. "Contextualized ranking of entity types based on knowledge graphs". In: *J. Web Semant.* 37-38 (2016), pp. 170–183. DOI: 10.1016/j.websem.2015.12.005 (cit. on p. 48).

[181] J. Waitelonis, C. Exeler, and H. Sack. "Linked data enabled generalized vector space model to improve document retrieval". In: *Proceedings of the Third NLP&DBpedia Workshop (NLP & DBpedia 2015), co-located with the 14th International Semantic Web Conference 2015 (ISWC 2015)*. Ed. by H. Paulheim, M. van Erp, A. Filipowska, P. N. Mendes, and M. Brümmer. Vol. 1581. 4. Bethlehem, Pennsylvania, USA: CEUR, Oct. 2015, pp. 34–44. URL: http://ceur-ws.org/Vol-1581/paper4.pdf (cit. on p. 48).

[182] F. Ensan and E. Bagheri. "Document Retrieval Model Through Semantic Linking". In: *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM 2017, Cambridge, United Kingdom, February 6-10, 2017*. 2017, pp. 181–190. DOI: 10.1145/3018661.3018692 (cit. on p. 48).

[183] M. Ciglan, K. Nørvåg, and L. Hluchý. "The SemSets model for ad-hoc semantic list search". In: *Proceedings of the 21st World Wide Web Conference 2012, WWW 2012, Lyon, France, April 16-20, 2012*. 2012, pp. 131–140. DOI: 10.1145/2187836.2187855 (cit. on p. 48).

[184] F. Hasibi, F. Nikolaev, C. Xiong, K. Balog, S. E. Bratsberg, A. Kotov, and J. Callan. "DBpedia-Entity v2: A Test Collection for Entity Search". In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*. 2017, pp. 1265–1268. DOI: 10.1145/3077136.3080751 (cit. on pp. 48, 72).

[185] Y. Gao, J. Liang, B. Han, M. Yakout, and A. Mohamed. *KDD Tutorial T39: Building a Large-scale, Accurate and Fresh Knowledge Graph*. https://kdd2018tutorialt39.azurewebsites.net/. Accessed on 2019-05-16. Aug. 2018 (cit. on pp. 49, 71, 334).

[186] A. Sinha, Z. Shen, Y. Song, H. Ma, D. Eide, B. P. Hsu, and K. Wang. "An Overview of Microsoft Academic Service (MAS) and Applications". In: *Proceedings of the 24th International Conference on World Wide Web Companion, WWW 2015, Florence, Italy, May 18-22, 2015 - Companion Volume*. Ed. by A. Gangemi, S. Leonardi, and A. Panconesi. ACM, 2015, pp. 243–246. DOI: 10.1145/2740908.2742839 (cit. on pp. 49, 71, 334).

[187] I. Bordino, Y. Mejova, and M. Lalmas. "Penguins in sweaters, or serendipitous entity search on user-generated content". In: *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management (CIKM 2013)*. 2013, pp. 109–118. ISBN: 9781450322638. DOI: 10.1145/2505515.2505680 (cit. on pp. 49, 71, 334).

[188] R. Baraglia, G. De Francisci Morales, and C. Lucchese. "Document similarity self-join with MapReduce". In: *2010 IEEE 10th International Conference on Data Mining (ICDM 2010)*. 2010, pp. 731–736. ISBN: 9780769542560. DOI: 10.1109/ICDM.2010.70 (cit. on p. 49).

[189] Y. Ni, Q. K. Xu, F. Cao, Y. Mass, D. Sheinwald, H. J. Zhu, and S. S. Cao. "Semantic Documents Relatedness using Concept Graph Representation". In: *Proceedings of the 9th ACM International Conference on Web Search and Data Mining - WSDM '16*. New York, New York, USA: ACM Press, 2016, pp. 635–644. ISBN: 9781450337168. DOI: 10.1145/2835776.2835801 (cit. on pp. 50, 71, 334).

[190] J. F. Sowa. *Conceptual structures: Information processing in mind and machine*. Addison-Wesley, 1984. ISBN: 0-201-14472-7 (cit. on pp. 50, 51).

[191] E. Yeh, D. Ramage, C. D. Manning, E. Agirre, and A. Soroa. "WikiWalk: Random walks on Wikipedia for Semantic Relatedness". In: *Proceedings of the 2009 Workshop on Graph-based Methods for Natural Language Processing, August*

*7, 2009, Singapore*. The Association for Computer Linguistics, 2009, pp. 41–49. URL: https://www.aclweb.org/anthology/W09-3206/ (cit. on p. 50).

[192] A. Huang, D. N. Milne, E. Frank, and I. H. Witten. "Learning a concept-based document similarity measure". In: *Journal of the Association for Information Science and Technology* 63.8 (2012), pp. 1593–1608. DOI: 10.1002/asi.22689 (cit. on p. 50).

[193] M. Nickel, V. Tresp, and H. Kriegel. "A Three-Way Model for Collective Learning on Multi-Relational Data". In: *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*. 2011, pp. 809–816. URL: https://icml.cc/2011/papers/438_icmlpaper.pdf (cit. on p. 50).

[194] G. H. L. Fletcher, J. Hidders, and J. L. Larriba-Pey, eds. *Graph Data Management, Fundamental Issues and Recent Developments*. Data-Centric Systems and Applications. Springer, 2018. ISBN: 978-3-319-96192-7. DOI: 10.1007/978-3-319-96193-4 (cit. on p. 51).

[195] J. Li, L. Zhang, and Y. Yu. "Learning to Generate Semantic Annotation for Domain Specific Sentences". In: *Proceedings of the K-CAP 2001 Workshop on Knowledge Markup and Semantic Annotation Victoria, B.C., Canada, October 21, 2001*. 2001. URL: http://ceur-ws.org/Vol-99/Jianming_Li-et-al.pdf (cit. on p. 51).

[196] E. Minkov and W. W. Cohen. "Improving graph-walk-based similarity with reranking: Case studies for personal information management". In: *ACM Trans. Inf. Syst.* 29.1 (2010), 4:1–4:52. DOI: 10.1145/1877766.1877770 (cit. on pp. 52, 71, 73, 334).

[197] W. W. Cohen, P. Ravikumar, and S. E. Fienberg. "A Comparison of String Distance Metrics for Name-Matching Tasks". In: *Proceedings of IJCAI-03 Workshop on Information Integration on the Web (IIWeb-03), August 9-10, 2003, Acapulco, Mexico*. Ed. by S. Kambhampati and C. A. Knoblock. 2003, pp. 73–78. URL: http://www.isi.edu/info-agents/workshops/ijcai03/papers/Cohen-p.pdf (cit. on p. 52).

[198] M. Zhong and M. Liu. "Ranking the answer trees of graph search by both structure and content". In: *Proceedings of the 1st Joint International Workshop on Entity-Oriented and Semantic Search*. Portland, OR, USA: Association for Computing Machinery, New York, NY, USA, Aug. 2012, pp. 1–3. DOI: 10.1145/2379307.2379314 (cit. on pp. 52, 71, 334).

[199] Y. Zhu, E. Yan, and I. Song. "A natural language interface to a graph-based bibliographic information retrieval system". In: *Data Knowl. Eng.* 111 (2017), pp. 73–89. DOI: 10.1016/j.datak.2017.06.006 (cit. on pp. 52, 71, 334).

[200] Z. Zhang, L. Wang, X. Xie, and H. Pan. "A Graph Based Document Retrieval Method". In: *22nd IEEE International Conference on Computer Supported Cooperative Work in Design, CSCWD 2018, Nanjing, China, May 9-11, 2018*. 2018, pp. 426–432. DOI: 10.1109/CSCWD.2018.8465295 (cit. on p. 53).

[201] X. Ouvrard, J. L. Goff, and S. Marchand-Maillet. "Adjacency and Tensor Representation in General Hypergraphs.Part 2: Multisets, Hb-graphs and Related e-adjacency Tensors". In: *CoRR* abs/1805.11952 (2018). arXiv: 1805.11952 (cit. on pp. 53, 73, 240, 252, 258).

[202] A. Frank, T. Király, and Z. Király. "On the orientation of graphs and hypergraphs". In: *Discrete Applied Mathematics* 131.2 (2003), pp. 385–400. DOI: 10.1016/S0166-218X(02)00462-6 (cit. on p. 53).

[203] L. M. Garshol. "Metadata? Thesauri? Taxonomies? Topic Maps! Making Sense of it all". In: *J. Information Science* 30.4 (2004), pp. 378–391. DOI: 10.1177/0165551504045856 (cit. on pp. 54, 71, 334).

[204] M. Yi. "Information organization and retrieval using a topic maps-based ontology: Results of a task-based evaluation". In: *JASIST* 59.12 (2008), pp. 1898–1911. DOI: 10.1002/asi.20899 (cit. on pp. 54, 71, 334).

[205] T. Menezes and C. Roth. "Semantic Hypergraphs". In: *CoRR* abs/1908.10784 (2019). arXiv: 1908.10784. URL: http://arxiv.org/abs/1908.10784 (cit. on pp. 54, 73).

[206] S. Xiong and D. Ji. "Query-focused multi-document summarization using hypergraph-based ranking". In: *Inf. Process. Manage.* 52.4 (2016), pp. 670–681. DOI: 10.1016/j.ipm.2015.12.012 (cit. on pp. 55, 335).

[207] C. Cattuto, C. Schmitz, A. Baldassarri, V. D. P. Servedio, V. Loreto, A. Hotho, M. Grahl, and G. Stumme. "Network properties of folksonomies". In: *AI Commun.* 20.4 (2007), pp. 245–262. URL: http://content.iospress.com/articles/ai-communications/aic410 (cit. on pp. 55, 335).

[208] S. B. Seidman. "Structures induced by collections of subsets: a hypergraph approach". In: *Mathematical Social Sciences* 1.4 (1981), pp. 381–396. DOI: 10.1016/0165-4896(81)90016-0 (cit. on p. 55).

[209] S. Tan, J. Bu, C. Chen, B. Xu, C. Wang, and X. He. "Using rich social media information for music recommendation via hypergraph model". In: *TOMCCAP* 7.Supplement (2011), p. 22. DOI: 10.1145/2037676.2037679 (cit. on pp. 55, 335).

[210] B. McFee and G. R. G. Lanckriet. "Hypergraph Models of Playlist Dialects". In: *Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR 2012, Mosteiro S.Bento Da Vitória, Porto, Portugal, October 8-12, 2012*. 2012, pp. 343–348. URL: http://ismir2012.ismir.net/event/papers/343-ismir-2012.pdf (cit. on pp. 55, 335).

[211] A. Theodoridis, C. Kotropoulos, and Y. Panagakis. "Music recommendation using hypergraphs and group sparsity". In: *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013*. 2013, pp. 56–60. DOI: 10.1109/ICASSP.2013.6637608 (cit. on pp. 55, 335).

[212] A. Borici and A. Thomo. "Semantic Graph Compression with Hypergraphs". In: *28th IEEE International Conference on Advanced Information Networking and Applications, AINA 2014, Victoria, BC, Canada, May 13-16, 2014*. 2014, pp. 1097–1104. DOI: 10.1109/AINA.2014.133 (cit. on p. 55).

[213] H. Lee-Kwang and K. Lee. "Fuzzy hypergraph and fuzzy partition". In: *IEEE Trans. Systems, Man, and Cybernetics* 25.1 (1995), pp. 196–201. DOI: 10.1109/21.362951 (cit. on pp. 55, 165, 252, 335).

[214] M. Akram and W. A. Dudek. "Intuitionistic fuzzy hypergraphs with applications". In: *Inf. Sci.* 218 (2013), pp. 182–193. DOI: 10.1016/j.ins.2012.06.024 (cit. on pp. 55, 252, 335).

[215] U. Chitra and B. J. Raphael. "Random Walks on Hypergraphs with Edge-Dependent Vertex Weights". In: *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*. 2019, pp. 1172–1181. URL: http://proceedings.mlr.press/v97/chitra19a.html (cit. on p. 55).

[216] G. Canfora and L. Cerulo. "A taxonomy of information retrieval models and tools". In: *Journal of Computing and Information Technology* 12.3 (2004), pp. 175–194. DOI: 10.2498/cit.2004.03.01 (cit. on p. 55).

[217] D. Harel. "On Visual Formalisms". In: *Commun. ACM* 31.5 (1988), pp. 514–530. DOI: 10.1145/42411.42414 (cit. on p. 55).

[218] A. Akhmediyarova, J. Kuandykova, B. Kubekov, I. T. Utepbergenov, and V. Popkov. "Objective of modeling and computation of city electric transportation networks properties". In: *Proc. of the Int. Conf. on Information Science and Management Engineering, Destech Publications, Inc.* 2015, pp. 106–111 (cit. on p. 55).

[219] J. Johnson. *Hypernetworks in the Science of Complex Systems*. Vol. 3. Series on Complexity Science. World Scientific, 2014. ISBN: 978-1-86094-972-2. DOI: 10.1142/p533 (cit. on p. 55).

[220] A. Basu and R. W. Blanning. "Metagraphs: A tool for modeling decision support systems". In: *Management Science* 40.12 (Dec. 1994), pp. 1579–1600. URL: https://www.jstor.org/stable/2632940 (cit. on pp. 55, 252).

[221] J. Huang, C. Chen, F. Ye, J. Wu, Z. Zheng, and G. Ling. "Hyper2vec: Biased Random Walk for Hyper-network Embedding". In: *Database Systems for Advanced Applications - DASFAA 2019 International Workshops: BDMS, BDQM, and GDMA, Chiang Mai, Thailand, April 22-25, 2019, Proceedings*. 2019, pp. 273–277. DOI: 10.1007/978-3-030-18590-9_27 (cit. on p. 55).

[222] L. Lovász et al. "Random walks on graphs: A survey". In: *Combinatorics, Paul erdos is eighty* 2.1 (1993), pp. 1–46 (cit. on pp. 56, 300).

[223] J. Leskovec and C. Faloutsos. "Sampling from large graphs". In: *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006*. 2006, pp. 631–636. DOI: 10.1145/1150402.1150479 (cit. on pp. 56, 300).

[224] A. D. Sarma, D. Nanongkai, G. Pandurangan, and P. Tetali. "Distributed Random Walks". In: *J. ACM* 60.1 (2013), 2:1–2:31. DOI: 10.1145/2432622.2432624 (cit. on p. 56).

[225] P. Pons and M. Latapy. "Computing Communities in Large Networks Using Random Walks". In: *Journal of Graph Algorithms and Applications* 10.2 (2006), pp. 191–218. URL: http://jgaa.info/accepted/2006/PonsLatapy2006.10.2.pdf (cit. on p. 56).

[226] Z. Guo and D. Barbosa. "Robust Entity Linking via Random Walks". In: *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014*. 2014, pp. 499–508. DOI: 10.1145/2661829.2661887 (cit. on p. 56).

[227] W. Shen, J. Wang, and J. Han. "Entity Linking with a Knowledge Base: Issues, Techniques, and Solutions". In: *IEEE Trans. Knowl. Data Eng.* 27.2 (2015), pp. 443–460. DOI: 10.1109/TKDE.2014.2327028 (cit. on pp. 56, 108).

[228] F. Chung. "A Brief Survey of PageRank Algorithms". In: *IEEE Trans. Network Science and Engineering* 1.1 (2014), pp. 38–42. DOI: 10.1109/TNSE.2014.2380315 (cit. on pp. 56, 300).

[229] T. Haveliwala. *Efficient Computation of PageRank*. Technical Report 1999-31. Stanford InfoLab, 1999. URL: http://ilpubs.stanford.edu:8090/386/ (cit. on pp. 56, 74).

[230] P. Berkhin. "A Survey on PageRank Computing". In: *Internet Mathematics* 2.1 (2005), pp. 73–120. DOI: 10.1080/15427951.2005.10129098 (cit. on pp. 56, 300).

[231] D. Gleich and L. Zhukov. *Scalable Computing for Power Law Graphs : Experience with Parallel PageRank*. Tech. rep. Yahoo! Research, 2005 (cit. on p. 56).

[232] C. Kohlschütter, P. Chirita, and W. Nejdl. "Efficient Parallel Computation of PageRank". In: *Advances in Information Retrieval, 28th European Conference on IR Research, ECIR 2006, London, UK, April 10-12, 2006, Proceedings*. 2006, pp. 241–252. DOI: 10.1007/11735106_22 (cit. on p. 56).

[233] J. R. Wicks and A. Greenwald. "More efficient parallel computation of PageRank". In: *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Amsterdam, The Netherlands, July 23-27, 2007*. 2007, pp. 861–862. DOI: 10.1145/1277741.1277946 (cit. on p. 56).

[234] B. Bahmani, A. Chowdhury, and A. Goel. "Fast Incremental and Personalized PageRank". In: *PVLDB* 4.3 (2010), pp. 173–184. DOI: 10.14778/1929861.1929864 (cit. on p. 56).

[235] Z. Nie, Y. Zhang, J. Wen, and W. Ma. "Object-level ranking: bringing order to Web objects". In: *Proceedings of the 14th International Conference on World Wide Web, WWW 2005, Chiba, Japan, May 10-14, 2005*. 2005, pp. 567–574. DOI: 10.1145/1060745.1060828 (cit. on pp. 57, 59, 299, 335).

[236] Z. Nie, J. Wen, and W. Ma. "Object-level Vertical Search". In: *CIDR 2007, Third Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 7-10, 2007, Online Proceedings*. 2007, pp. 235–246. URL: http://cidrdb.org/cidr2007/papers/cidr07p26.pdf (cit. on p. 57).

[237] D. Fogaras, B. Rácz, K. Csalogány, and T. Sarlós. "Towards Scaling Fully Personalized PageRank: Algorithms, Lower Bounds, and Experiments". In: *Internet Mathematics* 2.3 (2005), pp. 333–358. DOI: 10.1080/15427951.2005.10129104 (cit. on pp. 58, 257).

[238] C. Musto, G. Semeraro, M. de Gemmis, and P. Lops. "Tuning Personalized PageRank for Semantics-Aware Recommendations Based on Linked Open Data". In: *The Semantic Web - 14th International Conference, ESWC 2017, Portorož, Slovenia, May 28 - June 1, 2017, Proceedings, Part I*. 2017, pp. 169–183. DOI: 10.1007/978-3-319-58068-5_11 (cit. on pp. 58, 59, 71, 299, 305, 335).

[239] L. Espın-Noboa, F. Lemmerich, S. Walk, M. Strohmaier, and M. A. Musen. "HopRank: How Semantic Structure Influences Teleportation in PageRank (A Case Study on BioPortal)". In: *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*. 2019, pp. 2708–2714. DOI: 10.1145/3308558.3313487 (cit. on pp. 58, 59, 71, 299, 335).

[240] E. M. Voorhees. "TREC: Continuing information retrieval's tradition of experimentation". In: *Commun. ACM* 50.11 (2007), pp. 51–54. DOI: 10.1145/1297797.1297822 (cit. on pp. 60, 81).

[241] A. Komninos and A. Arampatzis. "Entity Ranking as a Search Engine Front-End". In: *International Journal On Advances in Internet Technology* 6.1 (June 2013), pp. 68–78. ISSN: 1942-2652. URL: http://www.thinkmind.org/index.php?view=article&articleid=inttech_v6_n12_2013_6 (cit. on p. 60).

[242] R. Blanco, H. Halpin, and D. Herzig. "Entity search evaluation over structured web data". In: *Proceedings of The First International Workshop on Entity-Oriented Search (EOS)*. 2011. URL: http://www.aifb.kit.edu/images/d/d9/EOS-SIGIR2011.pdf (cit. on p. 60).

[243] T. Tran, P. Mika, H. Wang, and M. Grobelnik. "SemSearch'11: the 4th semantic search workshop". In: *Proceedings of the 20th International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28 - April 1, 2011 (Companion Volume)*. 2011, pp. 315–316. DOI: 10.1145/1963192.1963329 (cit. on p. 60).

[244] S. Campinas, D. Ceccarelli, T. E. Perry, R. Delbru, K. Balog, and G. Tummarello. "The Sindice-2011 dataset for entity-oriented search in the web of data". In: *Proceedings of The First International Workshop on Entity-Oriented Search (EOS)*. 2011, pp. 26–32 (cit. on pp. 61, 72).

[245] F. Radlinski, M. Kurup, and T. Joachims. "How Does Clickthrough Data Reflect Retrieval Quality?" In: *Proceedings of the 17th ACM Conference on Information and Knowledge Management*. CIKM '08. Napa Valley, California, USA: ACM, 2008, pp. 43–52. ISBN: 978-1-59593-991-3. DOI: 10.1145/1458082.1458092 (cit. on pp. 64, 83, 98, 136, 156).

[246] K. Balog, L. Kelly, and A. Schuth. "Head First: Living Labs for Ad-hoc Search Evaluation". In: *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014*. Ed. by J. Li, X. S. Wang, M. N. Garofalakis, I. Soboroff, T. Suel, and M. Wang. ACM, 2014, pp. 1815–1818. DOI: 10.1145/2661829.2661962 (cit. on pp. 64, 72, 122, 127).

[247] G. Demartini, T. Iofciu, and A. P. de Vries. "Overview of the INEX 2009 Entity Ranking Track". In: *Focused Retrieval and Evaluation, 8th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2009, Brisbane, Australia, December 7-9, 2009, Revised and Selected Papers*. 2009, pp. 254–264. DOI: 10.1007/978-3-642-14556-8_26 (cit. on pp. 65, 88, 229, 231).

[248] P. Arvola, S. Geva, J. Kamps, R. Schenkel, A. Trotman, and J. Vainio. "Overview of the INEX 2010 Ad Hoc Track". In: *Comparative Evaluation of Focused Retrieval - 9th International Workshop of the Inititative for the Evaluation of XML Retrieval, INEX 2010, Vugh, The Netherlands, December 13-15, 2010, Revised Selected Papers*. 2010, pp. 1–32. DOI: 10.1007/978-3-642-23577-1_1 (cit. on pp. 66, 88, 93, 206, 229, 231).

[249] A. Sordoni, J. Nie, and Y. Bengio. "Modeling term dependencies with quantum language models for IR". In: *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '13, Dublin, Ireland - July 28 - August 01, 2013*. 2013, pp. 653–662. DOI: 10.1145/2484028.2484098 (cit. on p. 68).

[250] D. Aparicio, P. Ribeiro, and F. Silva. "Graphlet-orbit Transitions (GoT): A fingerprint for temporal network comparison". In: *PLoS One* 13 (10 Oct. 2018), e0205497. DOI: 10.1371/journal.pone.0205497 (cit. on pp. 69, 186).

[251] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. "TensorFlow: A System for Large-Scale Machine Learning". In: *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. Savannah, GA: USENIX Association, Nov. 2016, pp. 265–283. ISBN: 978-1-931971-33-1. URL: https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi (cit. on p. 69).

[252] P. Nikolov and V. Galabov. "Markov process simulation on a real quantum computer". In: *Proceedings of the 45th International Conference on Application of Mathematics in Engineering and Economics (AMEE 2019)* (2019). DOI: 10.1063/1.5133584 (cit. on p. 69).

[253] A. Louis. "Hypergraph Markov Operators, Eigenvalues and Approximation Algorithms". In: *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*. 2015, pp. 713–722. DOI: 10.1145/2746539.2746555 (cit. on p. 69).

[254] M. Allahyari. "Semantic Web Topic Models: Integrating Ontological Knowledge and Probabilistic Topic Models". PhD thesis. Athens, Georgia: University of Georgia, 2016 (cit. on pp. 71, 334).

[255] J. Allan, J. Callan, K. Collins-Thompson, B. Croft, F. Feng, D. Fisher, J. Lafferty, L. Larkey, T. N. Truong, P. Ogilvie, et al. *The lemur toolkit for language modeling and information retrieval*. http://lemurproject.org. Accessed on 2012-01-25. 2003 (cit. on pp. 72, 122).

[256] J. P. Callan, W. B. Croft, and S. M. Harding. "The INQUERY Retrieval System". In: *Proceedings of the International Conference on Database and Expert Systems Applications*. Valencia, Spain, 1992, pp. 78–83 (cit. on pp. 72, 122, 124).

[257] C. Xiong, Z. Liu, J. Callan, and E. H. Hovy. "JointSem: Combining Query Entity Linking and Entity based Document Ranking". In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017*. Ed. by E. Lim, M. Winslett, M. Sanderson, A. W. Fu, J. Sun, J. S. Culpepper, E. Lo, J. C. Ho, D. Donato, R. Agrawal, Y. Zheng, C. Castillo, A. Sun, V. S. Tseng, and C. Li. ACM, 2017, pp. 2391–2394. DOI: 10.1145/3132847.3133048 (cit. on pp. 72, 149, 331).

[258] C. Xiong, J. Callan, and T. Liu. "Word-Entity Duet Representations for Document Ranking". In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*. 2017, pp. 763–772. DOI: 10.1145/3077136.3080768 (cit. on pp. 72, 149).

[259] N. Toupikov, J. Umbrich, R. Delbru, M. Hausenblas, and G. Tummarello. "DING! Dataset Ranking using Formal Descriptions". In: *Proceedings of the WWW2009 Workshop on Linked Data on the Web, LDOW 2009, Madrid, Spain, April 20, 2009*. Ed. by C. Bizer, T. Heath, T. Berners-Lee, and K. Idehen. Vol. 538. CEUR Workshop Proceedings. CEUR-WS.org, 2009. URL: http://ceur-ws.org/Vol-538/ldow2009%5C_paper21.pdf (cit. on p. 72).

[260] H. Raviv, O. Kurland, and D. Carmel. "Document Retrieval Using Entity-Based Language Models". In: *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July 17-21, 2016*. 2016, pp. 65–74. DOI: 10.1145/2911451.2911508 (cit. on p. 72).

[261] O. Kurland and L. Lee. "PageRank without hyperlinks: structural re-ranking using links induced by language models". In: *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Salvador, Brazil, August 15-19, 2005*. 2005, pp. 306–313. DOI: 10.1145/1076034.1076087 (cit. on pp. 72, 305, 312).

[262] J. Hoffart, M. A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, and G. Weikum. "Robust Disambiguation of Named Entities in Text". In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*. ACL, 2011, pp. 782–792. URL: https://www.aclweb.org/anthology/D11-1072/ (cit. on pp. 72, 181).

[263] M. A. Yosef, J. Hoffart, I. Bordino, M. Spaniol, and G. Weikum. "AIDA: An Online Tool for Accurate Disambiguation of Named Entities in Text and

Tables". In: *Proc. VLDB Endow.* 4.12 (2011), pp. 1450–1453. URL: http://www.vldb.org/pvldb/vol4/p1450-yosef.pdf (cit. on p. 72).

[264] S. Elbassuoni, M. Ramanath, R. Schenkel, M. Sydow, and G. Weikum. "Language-model-based ranking for queries on RDF-graphs". In: *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM '09)*. 2009, pp. 977–986. ISBN: 9781605585123. DOI: 10.1145/1645953.1646078 (cit. on p. 72).

[265] S. E. Roberston. "The methodology of information retrieval experiment". In: *Information Retrieval Experiment* (1981), pp. 9–31 (cit. on p. 81).

[266] O. Arkhipova, L. Grauer, I. Kuralenok, and P. Serdyukov. "Search Engine Evaluation based on Search Engine Switching Prediction". In: *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015*. 2015, pp. 723–726. DOI: 10.1145/2766462.2767786 (cit. on p. 83).

[267] A. Culotta, A. McCallum, and J. Betz. "Integrating probabilistic extraction models and data mining to discover relations and patterns in text". In: *HLT-NAACL*. New York, NY, June 2006, pp. 296–303. URL: http://www.cs.umass.edu/~culotta/pubs/culotta06integrating.pdf (cit. on pp. 84, 99, 127).

[268] J. L. Devezas, C. T. Lopes, and S. Nunes. "FEUP at TREC 2017 OpenSearch Track Graph-Based Models for Entity-Oriented". In: *Proceedings of The Twenty-Sixth Text REtrieval Conference, TREC 2017, Gaithersburg, Maryland, USA, November 15-17, 2017*. Ed. by E. M. Voorhees and A. Ellis. Vol. 500-324. NIST Special Publication. National Institute of Standards and Technology (NIST), 2017. URL: https://trec.nist.gov/pubs/trec26/papers/FEUP-O.pdf (cit. on pp. 84, 155, 181, 337).

[269] J. Devezas and S. Nunes. "Graph-of-Entity: A Model for Combined Data Representation and Retrieval". In: *Proceedings of the 8th Symposium on Languages, Applications and Technologies (SLATE 2019)*. Vila do Conde, Portugal, 2019. DOI: 10.4230/OASIcs.SLATE.2019.1 (cit. on pp. 84, 299, 336).

[270] J. L. Devezas, S. Nunes, A. Guillén, Y. Gutiérrez, and R. Muñoz. "FEUP at TREC 2018 Common Core Track - Reranking for Diversity using Hypergraph-of-Entity and Document Profiling". In: *Proceedings of the Twenty-Seventh Text REtrieval Conference, TREC 2018, Gaithersburg, Maryland, USA, November 14-16, 2018*. Ed. by E. M. Voorhees and A. Ellis. Vol. 500-331. NIST Special Publication. National Institute of Standards and Technology (NIST), 2018. URL: https://trec.nist.gov/pubs/trec27/papers/FEUP-CC.pdf (cit. on pp. 84, 221, 336).

[271] N. Fuhr. "Some Common Mistakes In IR Evaluation, And How They Can Be Avoided". In: *SIGIR Forum* 51.3 (2017), pp. 32–41. DOI: 10.1145/3190580.3190586 (cit. on p. 84).

[272] C. Buckley and E. M. Voorhees. "Evaluating evaluation measure stability". In: *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, July 24-28, 2000, Athens, Greece*. 2000, pp. 33–40. DOI: 10.1145/345508.345543 (cit. on p. 85).

[273] S. Robertson. "On GMAP: and other transformations". In: *Proceedings of the 2006 ACM CIKM International Conference on Information and Knowledge Management, Arlington, Virginia, USA, November 6-11, 2006*. 2006, pp. 78–83. DOI: 10.1145/1183614.1183630 (cit. on p. 85).

[274] K. Järvelin and J. Kekäläinen. "Cumulated gain-based evaluation of IR techniques". In: *ACM Trans. Inf. Syst.* 20.4 (2002), pp. 422–446. DOI: 10.1145/582415.582418 (cit. on p. 85).

[275] S. Geva, J. Kamps, M. Lehtonen, R. Schenkel, J. A. Thom, and A. Trotman. "Overview of the INEX 2009 Ad Hoc Track". In: *Focused Retrieval and Evaluation, 8th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2009, Brisbane, Australia, December 7-9, 2009, Revised and Selected Papers*. 2009, pp. 4–25. DOI: 10.1007/978-3-642-14556-8_4 (cit. on pp. 88, 93).

[276] R. Jagerman, K. Balog, P. Schaer, J. Schaible, N. Tavakolpoursaleh, and M. de Rijke. "Overview of TREC OpenSearch 2017". In: *Proceedings of The Twenty-Sixth Text REtrieval Conference, TREC 2017, Gaithersburg, Maryland, USA, November 15-17, 2017*. Ed. by E. M. Voorhees and A. Ellis. Vol. 500-324. NIST Special Publication. National Institute of Standards and Technology (NIST), 2017. URL: https://trec.nist.gov/pubs/trec26/papers/Overview-O.pdf (cit. on pp. 88, 127).

[277] J. Devezas and S. Nunes. *Simple English Wikipedia Link Graph with Clickstream Transitions 2018-12 [dataset]*. INESC TEC research data repository. Mar. 2019. DOI: 10.25747/83vk-zt74 (cit. on pp. 99, 336).

[278] D. Greene and P. Cunningham. "Practical solutions to the problem of diagonal dominance in kernel document clustering". In: *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*. 2006, pp. 377–384. DOI: 10.1145/1143844.1143892 (cit. on p. 100).

[279] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. "Knowledge vault: a web-scale approach to probabilistic knowledge fusion". In: *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*. Ed. by S. A. Macskassy, C. Perlich, J. Leskovec, W. Wang, and R. Ghani. ACM, 2014, pp. 601–610. DOI: 10.1145/2623330.2623623 (cit. on p. 108).

[280] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data". In: *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*. Ed. by C. E. Brodley and A. P. Danyluk. Morgan Kaufmann, 2001, pp. 282–289 (cit. on p. 108).

[281] K. Nebhi. "Named Entity Disambiguation using Freebase and Syntactic Parsing". In: *Proceedings of the First International Workshop on Linked Data for Information Extraction (LD4IE 2013) co-located with the 12th International Semantic Web Conference (ISWC 2013), Sydney, Australia, October 21, 2013*. Ed. by A. L. Gentile, Z. Zhang, C. d'Amato, and H. Paulheim. Vol. 1057. CEUR Workshop Proceedings. CEUR-WS.org, 2013. URL: http://ceur-ws.org/Vol-1057/Nebhi_LD4IE2013.pdf (cit. on p. 109).

[282] S. Bird, E. Klein, and E. Loper. *Natural Language Processing with Python*. O'Reilly Media, Inc., 2009. URL: http://www.nltk.org/book/ (cit. on p. 109).

[283] J. Nivre, J. Hall, and J. Nilsson. "MaltParser: A Data-Driven Parser-Generator for Dependency Parsing". In: *Proceedings of the Fifth International Conference on Language Resources and Evaluation, LREC 2006, Genoa, Italy, May 22-28, 2006*. Ed. by N. Calzolari, K. Choukri, A. Gangemi, B. Maegaard, J. Mariani, J. Odijk, and D. Tapias. European Language Resources Association (ELRA), 2006, pp. 2216–2219. URL: http://www.lrec-conf.org/proceedings/lrec2006/summaries/162.html (cit. on p. 109).

[284] P. Stenetorp, S. Pyysalo, G. Topic, T. Ohta, S. Ananiadou, and J. Tsujii. "BRAT: a Web-based Tool for NLP-Assisted Text Annotation". In: *EACL 2012, 13th*

*Conference of the European Chapter of the Association for Computational Linguistics, Avignon, France, April 23-27, 2012*. Ed. by W. Daelemans, M. Lapata, and L. Màrquez. The Association for Computer Linguistics, 2012, pp. 102–107. URL: https://www.aclweb.org/anthology/E12-2021/ (cit. on p. 109).

[285] R. Rodrigues, H. G. Oliveira, and P. Gomes. "LemPORT: a High-Accuracy Cross-Platform Lemmatizer for Portuguese". In: *3rd Symposium on Languages, Applications and Technologies*. Ed. by M. J. V. Pereira, J. P. Leal, and A. Simões. Vol. 38. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2014, pp. 267–274. ISBN: 978-3-939897-68-2. DOI: 10.4230/OASIcs.SLATE.2014.267 (cit. on p. 109).

[286] N. Cardoso. "REMBRANDT - Reconhecimento de Entidades Mencionadas Baseado em Relações e ANálise Detalhada do Texto". In: *Desafios na avaliação conjunta do reconhecimento de entidades mencionadas: O Segundo HAREM*. Linguateca, 2008, pp. 195–211 (cit. on p. 109).

[287] C. Mota and D. Santos. "Desafios na avaliação conjunta do reconhecimento de entidades mencionadas: O Segundo HAREM". In: *Linguateca* (2008). URL: http://www.linguateca.pt/LivroSegundoHAREM/ (cit. on pp. 109, 110).

[288] H. Cunningham, Y. Wilks, and R. J. Gaizauskas. "GATE-a General Architecture for Text Engineering". In: *16th International Conference on Computational Linguistics, Proceedings of the Conference, COLING 1996, Center for Sprogteknologi, Copenhagen, Denmark, August 5-9, 1996*. 1996, pp. 1057–1060. URL: https://www.aclweb.org/anthology/C96-2187/ (cit. on p. 109).

[289] H. Cunningham, V. Tablan, I. Roberts, M. A. Greenwood, and N. Aswani. "Information Extraction and Semantic Annotation for Multi-Paradigm Information Management". In: *Current Challenges in Patent Information Retrieval*. Ed. by M. Lupu, K. Mayer, J. Tait, and A. J. Trippe. Vol. 29. The Information Retrieval Series. Springer, 2011, pp. 307–327. DOI: 10.1007/978-3-642-19231-9_15 (cit. on p. 109).

[290] E. R. Fonseca and J. L. G. Rosa. "Mac-Morpho Revisited: Towards Robust Part-of-Speech Tagging". In: *Proceedings of the 9th Brazilian Symposium in Information and Human Language Technology, STIL 2013, Fortaleza, Brazil, October 21-23, 2013*. Brazilian Computer Society, 2013. URL: https://www.aclweb.org/anthology/W13-4811/ (cit. on p. 109).

[291] C. Freitas and S. Afonso. "Bíblia Florestal: Um manual lingüístico da Floresta Sintá(c)tica". In: *Linguateca* (2007). URL: http://linguateca.dei.uc.pt/Floresta/BibliaFlorestal/ (cit. on p. 109).

[292] D. Nadeau and S. Sekine. "A survey of named entity recognition and classification". In: *Lingvisticae Investigationes* 30.1 (Jan. 2007), pp. 3–26. DOI: 10.1075/li.30.1.03nad (cit. on p. 112).

[293] J. Liu, P. Pasupat, Y. Wang, S. Cyphers, and J. Glass. "Query understanding enhanced by hierarchical parsing structures". In: *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*. Dec. 2013, pp. 72–77. DOI: 10.1109/ASRU.2013.6707708 (cit. on pp. 116, 257).

[294] J. Guo, G. Xu, X. Cheng, and H. Li. "Named entity recognition in query". In: *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2009)*. 2009, pp. 267–274. ISBN: 9781605584836. DOI: 10.1145/1571941.1571989 (cit. on p. 116).

[295] R. Blanco, G. Ottaviano, and E. Meij. "Fast and Space-Efficient Entity Linking for Queries". In: *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining - WSDM '15*. 2015, pp. 179–188. ISBN: 9781450333177. DOI: 10.1145/2684822.2685317 (cit. on p. 116).

[296] N. Aggarwal and P. Buitelaar. "A System Description of Natural Language Query over DBpedia". In: *Proceedings of the Workshop on Interacting with Linked Data, Heraklion, Greece, May 28, 2012*. Ed. by C. Unger, P. Cimiano, V. López, E. Motta, P. Buitelaar, and R. Cyganiak. Vol. 913. CEUR Workshop Proceedings. CEUR-WS.org, 2012, pp. 96–99. URL: http://ceur-ws.org/Vol-913/08%5C_ILD2012.pdf (cit. on p. 116).

[297] C. Middleton and R. Baeza-Yates. *A comparison of open source search engines*. 2007. URL: https://web.archive.org/web/20160313072328/http://wrg.upf.edu/WRG/dctos/Middleton-Baeza.pdf (cit. on p. 122).

[298] F. Hasibi, K. Balog, D. Garigliotti, and S. Zhang. "Nordlys: A Toolkit for Entity-Oriented and Semantic Search". In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '17. Shinjuku, Tokyo, Japan: ACM, 2017, pp. 1289–1292. ISBN: 978-1-4503-5022-8. DOI: 10.1145/3077136.3084149 (cit. on p. 123).

[299] T. Strohman, D. Metzler, H. Turtle, and W. B. Croft. "Indri: A language model-based search engine for complex queries". In: *Proceedings of the International Conference on Intelligent Analysis*. Vol. 2. 6. 2005, pp. 2–6. URL: https://web.archive.org/web/20080807192836/https://analysis.mitre.org/proceedings/Final_Papers_Files/150_Camera_Ready_Paper.pdf (cit. on p. 124).

[300] E. Gabrilovich, M. Ringgaard, and A. Subramanya. *FACC1: Freebase annotation of ClueWeb corpora, Version 1 (Release date 2013-06-26, Format version 1, Correction level 0)*. http://lemurproject.org/clueweb09/FACC1/. 2013 (cit. on p. 124).

[301] T. Ichimura, T. Uemoto, A. Hara, and K. J. Mackin. "Emergence of altruism behavior in army ant-based social evolutionary system". In: *Springerplus* 3 (Oct. 2014), p. 712. DOI: 10.1186/2193-1801-3-712 (cit. on p. 125).

[302] J. Devezas and S. Nunes. "Graph-Based Entity-Oriented Search: Imitating the Human Process of Seeking and Cross Referencing Information". In: *ERCIM News. Special Issue: Digital Humanities* 111 (Oct. 2017), pp. 13–14. URL: https://ercim-news.ercim.eu/en111/special/graph-based-entity-oriented-search-imitating-the-human-process-of-seeking-and-cross-referencing-information (cit. on pp. 125, 129, 149, 337).

[303] J. Devezas and S. Nunes. "Hypergraph-of-entity: A unified representation model for the retrieval of text and knowledge". In: *Open Computer Science* 9.1 (June 2019), pp. 103–127. DOI: 10.1515/comp-2019-0006 (cit. on pp. 129, 206, 336).

[304] G. A. Miller. "WordNet: A Lexical Database for English". In: *Commun. ACM* 38.11 (1995), pp. 39–41. DOI: 10.1145/219717.219748 (cit. on pp. 133, 196).

[305] A. Inselberg. *Parallel Coordinates: Visual Multidimensional Geometry and Its Applications*. Springer, 2009. ISBN: 978-0-470-85618-5 (cit. on p. 136).

[306] V. Tablan, D. Damljanovic, and K. Bontcheva. "A Natural Language Query Interface to Structured Information". In: *The Semantic Web: Research and Applications, 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008, Proceedings*. 2008, pp. 361–375. DOI: 10.1007/978-3-540-68234-9_28 (cit. on p. 148).

[307] M. S. Bernstein, J. Teevan, S. T. Dumais, D. J. Liebling, and E. Horvitz. "Direct answers for search queries in the long tail". In: *CHI Conference on Human Factors in Computing Systems, CHI '12, Austin, TX, USA - May 05 - 10, 2012*. 2012, pp. 237–246. DOI: 10.1145/2207676.2207710 (cit. on p. 149).

[308] F. Lashkari, F. Ensan, E. Bagheri, and A. A. Ghorbani. "Efficient indexing for semantic search". In: *Expert Syst. Appl.* 73 (2017), pp. 92–114. DOI: 10.1016/j.eswa.2016.12.033 (cit. on p. 162).

[309] D. Widdows and B. Dorow. "A Graph Model for Unsupervised Lexical Acquisition". In: *19th International Conference on Computational Linguistics, COLING 2002, Howard International House and Academia Sinica, Taipei, Taiwan, August 24 - September 1, 2002*. 2002. URL: https://www.aclweb.org/anthology/C02-1114/ (cit. on p. 162).

[310] R. F. i Cancho. "The structure of syntactic dependency networks: insights from recent advances in network theory". In: *Problems of quantitative linguistics* (2005), pp. 60–75 (cit. on p. 162).

[311] A. Z. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. L. Wiener. "Graph structure in the Web". In: *Comput. Networks* 33.1-6 (2000), pp. 309–320. DOI: 10.1016/S1389-1286(00)00083-9 (cit. on p. 162).

[312] Y. Guo, W. Che, T. Liu, and S. Li. "A Graph-based Method for Entity Linking". In: *Fifth International Joint Conference on Natural Language Processing, IJCNLP 2011, Chiang Mai, Thailand, November 8-13, 2011*. The Association for Computer Linguistics, 2011, pp. 1010–1018. URL: https://www.aclweb.org/anthology/I11-1113/ (cit. on p. 162).

[313] J. Hu, G. Wang, F. H. Lochovsky, J. Sun, and Z. Chen. "Understanding user's query intent with wikipedia". In: *Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, 2009*. Ed. by J. Quemada, G. León, Y. S. Maarek, and W. Nejdl. ACM, 2009, pp. 471–480. DOI: 10.1145/1526709.1526773 (cit. on p. 162).

[314] F. Gomes, J. L. Devezas, and Á. Figueira. "Temporal Visualization of a Multidimensional Network of News Clips". In: *Advances in Information Systems and Technologies [WorldCIST'13, Olhão, Algarve, Portugal, March 27-30, 2013]*. Ed. by Á. Rocha, A. M. R. Correia, T. Wilson, and K. A. Stroetmann. Vol. 206. Advances in Intelligent Systems and Computing. Springer, 2013, pp. 157–166. DOI: 10.1007/978-3-642-36981-0_15 (cit. on p. 164).

[315] A. Banerjee and A. Char. *On the spectrum of directed uniform and non-uniform hypergraphs*. Oct. 2017. arXiv: 1710.06367 [math.SP] (cit. on p. 166).

[316] S. R. Gallagher and D. S. Goldberg. "Clustering Coefficients in Protein Interaction Hypernetworks". In: *ACM Conference on Bioinformatics, Computational Biology and Biomedical Informatics. ACM-BCB 2013, Washington, DC, USA, September 22-25, 2013*. 2013, p. 552. DOI: 10.1145/2506583.2506635 (cit. on pp. 167, 189, 191).

[317] T. H. Nelson. "Complex Information Processing: A File Structure for the Complex, the Changing and the Indeterminate". In: *Proceedings of the 1965 20th National Conference*. ACM '65. Cleveland, Ohio, USA: Association for Computing Machinery, 1965, pp. 84–100. ISBN: 9781450374958. DOI: 10.1145/800197.806036 (cit. on p. 169).

[318] J. Conklin. "Hypertext: An Introduction and Survey". In: *Computer* 20.9 (1987), pp. 17–41. DOI: 10.1109/MC.1987.1663693 (cit. on p. 169).

[319] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. "The Web as a Graph". In: *Proceedings of the Nineteenth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, May 15-17, 2000, Dallas, Texas, USA*. Ed. by V. Vianu and G. Gottlob. ACM, 2000, pp. 1–10. DOI: 10.1145/335168.335170 (cit. on p. 169).

[320] J. F. Sowa. "Conceptual Graphs for a Data Base Interface". In: *IBM J. Res. Dev.* 20.4 (1976), pp. 336–357. DOI: 10.1147/rd.204.0336 (cit. on p. 169).

[321] S. Robertson. "Understanding inverse document frequency: on theoretical arguments for IDF". In: *Journal of Documentation* 60.5 (2004), pp. 503–520. DOI: 10.1108/00220410410560582 (cit. on p. 178).

[322] A. Alhelbawy and R. J. Gaizauskas. "Graph Ranking for Collective Named Entity Disambiguation". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers*. The Association for Computer Linguistics, 2014, pp. 75–80. DOI: 10.3115/v1/p14-2013 (cit. on p. 181).

[323] E. Estrada and J. A. Rodriguez-Velazquez. *Complex Networks as Hypergraphs*. Tech. rep. physics/0505137. May 2005. URL: https://cds.cern.ch/record/836579 (cit. on p. 185).

[324] G. Csardi, T. Nepusz, et al. "The igraph software package for complex network research". In: *InterJournal, Complex Systems* 1695.5 (2006), pp. 1–9. URL: http://interjournal.org/manuscript_abstract.php?361100992 (cit. on p. 185).

[325] M. Bastian, S. Heymann, and M. Jacomy. "Gephi: An Open Source Software for Exploring and Manipulating Networks". In: *Proceedings of the Third International Conference on Weblogs and Social Media, ICWSM 2009, San Jose, California, USA, May 17-20, 2009*. 2009. URL: http://aaai.org/ocs/index.php/ICWSM/09/paper/view/154 (cit. on p. 185).

[326] M. Himsolt. *GML: A portable graph file format*. Tech. rep. Universität Passau, 1997 (cit. on p. 185).

[327] U. Brandes, M. Eiglsperger, I. Herman, M. Himsolt, and M. S. Marshall. "GraphML Progress Report Structural Layer Proposal". In: *Graph Drawing*. Ed. by P. Mutzel, M. Jünger, and S. Leipert. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 501–512. ISBN: 978-3-540-45848-7 (cit. on p. 185).

[328] E. M. Voorhees. "The Efficiency of Inverted Index and Cluster Searches". In: *Proceedings of the 9th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Pisa, Italy, September 8-10, 1986*. 1986, pp. 164–174. DOI: 10.1145/253168.253203 (cit. on pp. 186, 187).

[329] J. Zobel, A. Moffat, and K. Ramamohanarao. "Inverted Files Versus Signature Files for Text Indexing". In: *ACM Trans. Database Syst.* 23.4 (1998), pp. 453–490. DOI: 10.1145/296854.277632 (cit. on pp. 186, 187).

[330] H. Halpin. "A Query-Driven Characterization of Linked Data". In: *Proceedings of the WWW2009 Workshop on Linked Data on the Web, LDOW 2009, Madrid, Spain, April 20, 2009*. 2009. URL: http://ceur-ws.org/Vol-538/ldow2009%5C_paper16.pdf (cit. on p. 187).

[331] W. Ge, J. Chen, W. Hu, and Y. Qu. "Object Link Structure in the Semantic Web". In: *The Semantic Web: Research and Applications, 7th Extended Semantic Web Conference, ESWC 2010, Heraklion, Crete, Greece, May 30 - June 3, 2010, Proceedings, Part II*. 2010, pp. 257–271. DOI: 10.1007/978-3-642-13489-0_18 (cit. on p. 187).

[332] J. D. Fernández, M. A. Martínez-Prieto, P. de la Fuente Redondo, and C. Gutiérrez. "Characterising RDF data sets". In: *Journal of Information Science* 44.2 (2018), pp. 203–229. DOI: 10.1177/0165551516677945 (cit. on p. 187).

[333] P. Erdös. "On some extremal problems on r-graphs". In: *Discret. Math.* 1.1 (1971), pp. 1–6. DOI: 10.1016/0012-365X(71)90002-1 (cit. on p. 188).

[334] W. Brown, P. Erdos, and V. T Sós. "Some extremal problems on r-graphs". In: *New directions in the theory of graphs (Proc. Third Ann Arbor Conf., Univ. Michigan, Ann Arbor, Mich, 1971)*. New York, NY, USA: Academic Press, Inc., 1973, pp. 53–63. URL: http://real.mtak.hu/110443/ (cit. on p. 188).

[335] E. Sperner. "Ein satz über Untermengen einer endlichen Menge". In: *Mathematische Zeitschrift* 27.1 (1928), pp. 544–548. DOI: 10.1007/BF01171114 (cit. on p. 188).

[336] P. Turán. "On an extremal problem in graph theory". In: *Matematikai és Fizikai Lapok* 48 (1941), pp. 436–452. URL: https://ci.nii.ac.jp/naid/10022295281/ (cit. on p. 188).

[337] P. Turán. "Research problems". In: *Magyar Tud. Akad. Mat. Kutato Internat. Közl.* 6 (1961), pp. 417–423 (cit. on p. 188).

[338] P. Erdös, A. W. Goodman, and L. Pósa. "The representation of a graph by set intersections". In: *Canadian Journal of Mathematics* 18 (1966), pp. 106–112. DOI: 10.4153/CJM-1966-014-3 (cit. on p. 188).

[339] S. Klamt, U. Haus, and F. J. Theis. "Hypergraphs and Cellular Networks". In: *PLoS Computational Biology* 5.5 (2009). DOI: 10.1371/journal.pcbi.1000385 (cit. on pp. 188, 190).

[340] G. Gallo, G. Longo, and S. Pallottino. "Directed Hypergraphs and Applications". In: *Discret. Appl. Math.* 42.2 (1993), pp. 177–201. DOI: 10.1016/0166-218X(93)90045-P (cit. on p. 188).

[341] G. Ausiello, G. F. Italiano, and U. Nanni. *Optimal traversal of directed hypergraphs*. Tech. rep. ICSI Technical Report TR-92-073. International Computer Science Institute, 1992. URL: http://www.icsi.berkeley.edu/pubs/techreports/tr-92-073.pdf (cit. on p. 188).

[342] J. Gao, Q. Zhao, W. Ren, A. Swami, R. Ramanathan, and A. Bar-Noy. "Dynamic Shortest Path Algorithms for Hypergraphs". In: *IEEE/ACM Trans. Netw.* 23.6 (2015), pp. 1805–1817. DOI: 10.1109/TNET.2014.2343914 (cit. on p. 188).

[343] B. F. Ribeiro, P. Basu, and D. Towsley. "Multiple random walks to uncover short paths in power law networks". In: *2012 Proceedings IEEE INFOCOM Workshops, Orlando, FL, USA, March 25-30, 2012*. 2012, pp. 250–255. DOI: 10.1109/INFCOMW.2012.6193500 (cit. on pp. 188, 191).

[344] M. Głąbowski, B. Musznicki, P. Nowak, and P. Zwierzykowski. "Shortest Path Problem Solving Based on Ant Colony Optimization Metaheuristic". In: *Image Processing & Communications* 17.1-2 (Apr. 2012), pp. 7–17. DOI: 10.2478/v10248-012-0011-5 (cit. on p. 189).

[345] D. Li. "Shortest paths through a reinforced random walk". MA thesis. Department of Mathematics, Analysis and Applied Mathematics, University of Uppsala, 2011. URL: http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-153802 (cit. on p. 189).

[346] D. Mubayi and Y. Zhao. "Co-degree density of hypergraphs". In: *J. Comb. Theory, Ser. A* 114.6 (2007), pp. 1118–1132. DOI: 10.1016/j.jcta.2006.11.006 (cit. on p. 189).

[347] W. Yu and N. Sun. "Establishment and Analysis of the Supernetwork Model for Nanjing Metro Transportation System". In: *Complexity, Special Issue on Analysis and Applications of Complex Social Networks 2018* 2018.4860531 (2018), pp. 1–11. DOI: 10.1155/2018/4860531 (cit. on p. 190).

[348] D. J. Watts and S. H. Strogatz. "Collective dynamics of 'small-world' networks". In: *Nature* 393.6684 (June 1998), pp. 440–442. DOI: 10.1038/30918 (cit. on pp. 191, 194).

[349] L. Backstrom, P. Boldi, M. Rosa, J. Ugander, and S. Vigna. "Four degrees of separation". In: *Web Science 2012, WebSci '12, Evanston, IL, USA - June 22 - 24, 2012*. Ed. by N. S. Contractor, B. Uzzi, M. W. Macy, and W. Nejdl. ACM, 2012, pp. 33–42. DOI: 10.1145/2380718.2380723 (cit. on pp. 193, 198).

[350] S. Milgram. "The small world problem". In: *Psychology Today* 2.1 (1967), pp. 60–67 (cit. on p. 198).

[351] J. Travers and S. Milgram. "An experimental study of the small world problem". In: *Social Networks: A Developing Paradigm*. Ed. by S. Leinhardt. Washington, DC: Academic Press, 1977, pp. 179–197. ISBN: 978-0-12-442450-0. DOI: 10.1016/B978-0-12-442450-0.50018-3 (cit. on p. 198).

[352] A. V. Aho and M. J. Corasick. "Efficient String Matching: An Aid to Bibliographic Search". In: *Commun. ACM* 18.6 (1975), pp. 333–340. DOI: 10.1145/360825.360855 (cit. on p. 222).

[353] M. Abualsaud, G. V. Cormack, N. Ghelani, A. Ghenai, M. R. Grossman, S. Rahbariasl, H. Zhang, and M. D. Smucker. "UWaterlooMDS at the TREC 2018 Common Core Track". In: *Proceedings of the Twenty-Seventh Text REtrieval Conference, TREC 2018, Gaithersburg, Maryland, USA, November 14-16, 2018*. Ed. by E. M. Voorhees and A. Ellis. Vol. 500-331. NIST Special Publication. National Institute of Standards and Technology (NIST), 2018. URL: https://trec.nist.gov/pubs/trec27/papers/UWaterlooMDS-CC.pdf (cit. on p. 224).

[354] R. Benham, L. Gallagher, J. Mackenzie, B. Liu, X. Lu, F. Scholer, J. S. Culpepper, and A. Moffat. "RMIT at the 2018 TREC CORE Track". In: *Proceedings of the Twenty-Seventh Text REtrieval Conference, TREC 2018, Gaithersburg, Maryland, USA, November 14-16, 2018*. Ed. by E. M. Voorhees and A. Ellis. Vol. 500-331. NIST Special Publication. National Institute of Standards and Technology (NIST), 2018. URL: https://trec.nist.gov/pubs/trec27/papers/RMIT-CC.pdf (cit. on p. 224).

[355] R. Yu, Y. Xie, and J. Lin. "H2oloo at TREC 2018: Cross-Collection Relevance Transfer for the Common Core Track". In: *Proceedings of the Twenty-Seventh Text REtrieval Conference, TREC 2018, Gaithersburg, Maryland, USA, November 14-16, 2018*. Ed. by E. M. Voorhees and A. Ellis. Vol. 500-331. NIST Special Publication. National Institute of Standards and Technology (NIST), 2018. URL: https://trec.nist.gov/pubs/trec27/papers/h2oloo-CC.pdf (cit. on p. 224).

[356] M. R. Grossman and G. V. Cormack. "MRG_UWaterloo Participation in the TREC 2018 Common Core Track". In: *Proceedings of the Twenty-Seventh Text REtrieval Conference, TREC 2018, Gaithersburg, Maryland, USA, November 14-16, 2018*. Ed. by E. M. Voorhees and A. Ellis. Vol. 500-331. NIST Special Publication. National Institute of Standards and Technology (NIST), 2018. URL: https://trec.nist.gov/pubs/trec27/papers/MRG%5C_UWaterloo-CC.pdf (cit. on p. 224).

[357] P. Yang and J. Lin. "Anserini at TREC 2018: CENTRE, Common Core, and News Tracks". In: *Proceedings of the Twenty-Seventh Text REtrieval Conference, TREC 2018, Gaithersburg, Maryland, USA, November 14-16, 2018*. Ed. by E. M. Voorhees and A. Ellis. Vol. 500-331. NIST Special Publication. National Institute of Standards and Technology (NIST), 2018. URL: https://trec.nist.gov/pubs/trec27/papers/anserini-CTR-CC-N.pdf (cit. on pp. 224, 248).

[358] G. Gonçalves, J. Magalhães, C. Xiong, and J. Callan. "Improving Ad Hoc Retrieval With Bag Of Entities". In: *Proceedings of the Twenty-Seventh Text REtrieval Conference, TREC 2018, Gaithersburg, Maryland, USA, November 14-16, 2018*. Ed. by E. M. Voorhees and A. Ellis. Vol. 500-331. NIST Special Publication. National Institute of Standards and Technology (NIST), 2018. URL: https://trec.nist.gov/pubs/trec27/papers/NOVASearch-CC.pdf (cit. on p. 224).

[359] S. Naseri, J. Foley, and J. Allan. "UMass at TREC 2018: CAR, Common Core and News Tracks". In: *Proceedings of the Twenty-Seventh Text REtrieval Conference, TREC 2018, Gaithersburg, Maryland, USA, November 14-16, 2018*. Ed. by E. M. Voorhees and A. Ellis. Vol. 500-331. NIST Special Publication. National Institute of Standards and Technology (NIST), 2018. URL: https://trec.nist.gov/pubs/trec27/papers/UMass-CAR-CC-N.pdf (cit. on p. 224).

[360] B. Aklouche, I. Bounhas, and Y. Slimani. "Query Expansion Based on NLP and Word Embeddings". In: *Proceedings of the Twenty-Seventh Text REtrieval Conference, TREC 2018, Gaithersburg, Maryland, USA, November 14-16, 2018*. Ed. by E. M. Voorhees and A. Ellis. Vol. 500-331. NIST Special Publication. National Institute of Standards and Technology (NIST), 2018. URL: https://trec.nist.gov/pubs/trec27/papers/JARIR-CC.pdf (cit. on p. 224).

[361] A. Bondarenko, M. Hagen, M. Völske, B. Stein, A. Panchenko, and C. Biemann. "Webis at TREC 2018: Common Core Track". In: *Proceedings of the Twenty-Seventh Text REtrieval Conference, TREC 2018, Gaithersburg, Maryland, USA, November 14-16, 2018*. Ed. by E. M. Voorhees and A. Ellis. Vol. 500-331. NIST Special Publication. National Institute of Standards and Technology (NIST), 2018. URL: https://trec.nist.gov/pubs/trec27/papers/Webis-CC.pdf (cit. on p. 224).

[362] R. Campos, V. Mangaravite, A. Pasquali, A. M. Jorge, C. Nunes, and A. Jatowt. "A Text Feature Based Automatic Keyword Extraction Method for Single Documents". In: *Advances in Information Retrieval - 40th European Conference on IR Research, ECIR 2018, Grenoble, France, March 26-29, 2018, Proceedings*. 2018, pp. 684–691. DOI: 10.1007/978-3-319-76941-7_63 (cit. on pp. 226, 227).

[363] R. Mihalcea and P. Tarau. "TextRank: Bringing Order into Text". In: *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing , EMNLP 2004, A meeting of SIGDAT, a Special Interest Group of the ACL, held in conjunction with ACL 2004, 25-26 July 2004, Barcelona, Spain*. 2004, pp. 404–411. URL: https://www.aclweb.org/anthology/W04-3252/ (cit. on pp. 226, 227, 251).

[364] X. Wan and J. Xiao. "Single Document Keyphrase Extraction Using Neighborhood Knowledge". In: *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*. 2008, pp. 855–860. URL: http://www.aaai.org/Library/AAAI/2008/aaai08-136.php (cit. on pp. 226, 227).

[365] S. Rose, D. Engel, N. Cramer, and W. Cowley. "Automatic Keyword Extraction from Individual Documents". In: *Text Mining*. John Wiley & Sons, Ltd, 2010. Chap. 1, pp. 1–20. ISBN: 9780470689646. DOI: 10.1002/9780470689646.ch1 (cit. on pp. 226, 227).

[366] J. Devezas. "Graph-Based Entity-Oriented Search: A Unified Framework in Information Retrieval". In: *Advances in Information Retrieval - 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14-17, 2020, Proceedings, Part II*. Ed. by J. M. Jose, E. Yilmaz, J. Magalhães, P. Castells, N. Ferro, M. J. Silva, and F. Martins. Vol. 12036. Lecture Notes in Computer Sci-

ence. Springer, 2020, pp. 602–607. DOI: 10.1007/978-3-030-45442-5_78 (cit. on pp. 229, 336).

[367] E. Yilmaz, E. Kanoulas, and J. A. Aslam. "A simple and efficient sampling method for estimating AP and NDCG". In: *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2008, Singapore, July 20-24, 2008*. 2008, pp. 603–610. DOI: 10.1145/1390334.1390437 (cit. on p. 231).

[368] D. Turnbull. *BM25 The Next Generation of Lucene Relevance*. https://opensourceconnections.com/blog/2015/10/16/bm25-the-next-generation-of-lucene-relevation/. Accessed on 2020-05-28. Oct. 2015 (cit. on p. 233).

[369] A. Guillén, Y. Gutiérrez, and R. Muñoz. "Natural Language Processing Technologies for Document Profiling". In: *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017, Varna, Bulgaria, September 2 - 8, 2017*. 2017, pp. 284–290. DOI: 10.26615/978-954-452-049-6_039 (cit. on p. 251).

[370] F. Coelho and C. Ribeiro. "Automatic illustration with cross-media retrieval in large-scale collections". In: *2011 9th International Workshop on Content-Based Multimedia Indexing (CBMI)*. June 2011, pp. 25–30. DOI: 10.1109/CBMI.2011.5972515 (cit. on pp. 257, 260).

[371] S. Hatakenaka and T. Miura. "Query and Topic Sensitive PageRank for general documents". In: *14th IEEE International Symposium on Web Systems Evolution, WSE 2012, Trento, Italy, September 28, 2012*. 2012, pp. 97–101. DOI: 10.1109/WSE.2012.6320539 (cit. on pp. 299, 308, 313).

[372] M. Richardson and P. M. Domingos. "The Intelligent surfer: Probabilistic Combination of Link and Content Information in PageRank". In: *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*. Ed. by T. G. Dietterich, S. Becker, and Z. Ghahramani. MIT Press, 2001, pp. 1441–1448. URL: http://papers.nips.cc/paper/2047-the-intelligent-surfer-probabilistic-combination-of-link-and-content-information-in-pagerank (cit. on pp. 299, 308, 313).

[373] M. A. Klopotek, S. T. Wierzchon, K. Ciesielski, D. Czerski, and M. Draminski. "Lazy Walks Versus Walks with Backstep: Flavor of PageRank". In: *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), Warsaw, Poland, August 11-14, 2014 - Volume II*. 2014, pp. 262–265. DOI: 10.1109/WI-IAT.2014.42 (cit. on p. 299).

[374] L. Tran, T. Quan, and A. Mai. "PageRank algorithm for Directed Hypergraph". In: *CoRR* abs/1909.01132 (2019). arXiv: 1909.01132 (cit. on p. 299).

[375] P. Li. "Learning on graphs with high-order relations: spectral methods, optimization and applications". PhD thesis. University of Illinois at Urbana-Champaign, June 2019. URL: http://hdl.handle.net/2142/105596 (cit. on p. 299).

[376] F. R. K. Chung, P. Horn, and J. Hughes. "Multi-commodity Allocation for Dynamic Demands Using PageRank Vectors". In: *Algorithms and Models for the Web Graph - 9th International Workshop, WAW 2012, Halifax, NS, Canada, June 22-23, 2012. Proceedings*. 2012, pp. 138–152. DOI: 10.1007/978-3-642-30541-2_11 (cit. on p. 300).

[377] V. Hristidis, Y. Wu, and L. Raschid. "Efficient Ranking on Entity Graphs with Personalized Relationships". In: *IEEE Transactions on Knowledge and Data Engineering* 26.4 (Apr. 2014), pp. 850–863. ISSN: 1041-4347. DOI: 10.1109/TKDE.2013.52 (cit. on p. 305).

[378] W. Xie, D. Bindel, A. J. Demers, and J. Gehrke. "Edge-Weighted Personalized PageRank: Breaking A Decade-Old Performance Barrier". In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*. 2015, pp. 1325–1334. DOI: 10.1145/2783258.2783278 (cit. on p. 305).

[379] K. Avrachenkov, N. Litvak, D. Nemirovsky, E. Smirnova, and M. Sokol. "Monte Carlo Methods for Top-k Personalized PageRank Lists and Name Disambiguation". In: *CoRR* abs/1008.3775 (2010). arXiv: 1008.3775 (cit. on p. 305).

[380] A. Pathak, S. Chakrabarti, and M. S. Gupta. "Index Design for Dynamic Personalized PageRank". In: *Proceedings of the 24th International Conference on Data Engineering, ICDE 2008, April 7-12, 2008, Cancún, Mexico*. 2008, pp. 1489–1491. DOI: 10.1109/ICDE.2008.4497599 (cit. on p. 305).

[381] G. Iván and V. Grolmusz. "When the Web meets the cell: using personalized PageRank for analyzing protein interaction networks". In: *Bioinformatics* 27.3 (2011), pp. 405–407. DOI: 10.1093/bioinformatics/btq680 (cit. on p. 305).

[382] H. Avron and L. Horesh. "Community Detection Using Time-Dependent Personalized PageRank". In: *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*. Ed. by F. Bach and D. Blei. Vol. 37. Lille, France: PMLR, July 2015, pp. 1795–1803. URL: http://jmlr.org/proceedings/papers/v37/avron15.html (cit. on p. 305).

[383] P. Lahoti, G. D. F. Morales, and A. Gionis. "Finding topical experts in Twitter via query-dependent personalized PageRank". In: *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017, Sydney, Australia, July 31 - August 03, 2017*. 2017, pp. 155–162. DOI: 10.1145/3110025.3110044 (cit. on p. 305).

[384] A. Anil, S. R. Singh, and R. Sarmah. "Mining heterogeneous terrorist attack network using personalized PageRank". In: *Web Intelligence* 16.1 (2018), pp. 37–52. DOI: 10.3233/WEB-180372 (cit. on p. 305).

[385] X. Wang, A. Shakery, and T. Tao. "Dirichlet PageRank". In: *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Salvador, Brazil, August 15-19, 2005*. 2005, pp. 661–662. DOI: 10.1145/1076034.1076178 (cit. on pp. 306, 312).

[386] C. Zhai and J. D. Lafferty. "A Study of Smoothing Methods for Language Models Applied to Ad Hoc Information Retrieval". In: *SIGIR Forum* 51.2 (2017), pp. 268–276. DOI: 10.1145/3130348.3130377 (cit. on p. 306).

[387] F. R. K. Chung, A. Tsiatas, and W. Xu. "Dirichlet PageRank and Trust-Based Ranking Algorithms". In: *Algorithms and Models for the Web Graph - 8th International Workshop, WAW 2011, Atlanta, GA, USA, May 27-29, 2011. Proceedings*. 2011, pp. 103–114. DOI: 10.1007/978-3-642-21286-4_9 (cit. on pp. 307, 312).

[388] F. Chung, A. Tsiatas, and W. Xu. "Dirichlet PageRank and Ranking Algorithms Based on Trust and Distrust". In: *Internet Mathematics* 9.1 (2013), pp. 113–134. DOI: 10.1080/15427951.2012.678194 (cit. on pp. 307, 312).

[389] T. H. Haveliwala. "Topic-sensitive PageRank". In: *Proceedings of the Eleventh International World Wide Web Conference, WWW 2002, May 7-11, 2002, Honolulu, Hawaii, USA*. 2002, pp. 517–526. DOI: 10.1145/511446.511513 (cit. on pp. 307, 313).

[390] S. Al-Saffar and G. L. Heileman. "Experimental Bounds on the Usefulness of Personalized and Topic-Sensitive PageRank". In: *2007 IEEE / WIC / ACM International Conference on Web Intelligence, WI 2007, 2-5 November 2007, Silicon*

*Valley, CA, USA, Main Conference Proceedings*. 2007, pp. 671–675. DOI: 10.1109/WI.2007.75 (cit. on p. 307).

[391] M. Rezvani and S. M. Hashemi. "Enhancing Accuracy of Topic Sensitive PageRank Using Jaccard Index and Cosine Similarity". In: *2012 IEEE/WIC/ACM International Conferences on Web Intelligence, WI 2012, Macau, China, December 4-7, 2012*. 2012, pp. 620–624. DOI: 10.1109/WI-IAT.2012.166 (cit. on pp. 308, 313).

[392] A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme. "Information Retrieval in Folksonomies: Search and Ranking". In: *The Semantic Web: Research and Applications, 3rd European Semantic Web Conference, ESWC 2006, Budva, Montenegro, June 11-14, 2006, Proceedings*. 2006, pp. 411–426. DOI: 10.1007/11762256_31 (cit. on pp. 309, 313).

[393] W. Li, D. Liu, M. K. Ng, and S. Vong. "The uniqueness of multilinear PageRank vectors". In: *Numerical Lin. Alg. with Applic.* 24.6 (2017). DOI: 10.1002/nla.2107 (cit. on p. 310).

[394] B. Meini and F. Poloni. "Perron-based algorithms for the multilinear PageRank". In: *Numerical Lin. Alg. with Applic.* 25.6 (2018). DOI: 10.1002/nla.2177 (cit. on p. 310).

[395] R. Pemantle. "Vertex-reinforced random walk". In: *Probability Theory and Related Fields* 92.1 (Mar. 1992), pp. 117–136. DOI: 10.1007/BF01205239 (cit. on p. 310).

[396] A. R. Benson, D. F. Gleich, and L. Lim. "The Spacey Random Walk: A Stochastic Process for Higher-Order Data". In: *SIAM Review* 59.2 (2017), pp. 321–345. DOI: 10.1137/16M1074023 (cit. on p. 310).

[397] A. R. Benson, D. F. Gleich, and J. Leskovec. "Tensor Spectral Clustering for Partitioning Higher-order Network Structures". In: *Proceedings of the 2015 SIAM International Conference on Data Mining, Vancouver, BC, Canada, April 30 - May 2, 2015*. 2015, pp. 118–126. DOI: 10.1137/1.9781611974010.14 (cit. on p. 310).

[398] T. Wu and D. F. Gleich. "Retrospective Higher-Order Markov Processes for User Trails". In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*. 2017, pp. 1185–1194. DOI: 10.1145/3097983.3098127 (cit. on p. 310).

[399] G. H. Golub and C. F. Van Loan. *Matrix computations*. Vol. 4. Johns Hopkins Studies in the Mathematical Sciences. The Johns Hopkins University Press, Feb. 2013. ISBN: 978-1-4214-0794-4 (cit. on p. 310).

[400] T. Dayar. "Analyzing Markov chains based on Kronecker products". In: *Markov Anniversary Meeting (MAM), An International Conference to celebrate the 150th Anniversary of the birth of A.A. Markov*. Ed. by A. N. Langville and W. J. Stewart. College of Charleston, SC, USA, 2006, pp. 279–300. URL: https://www.csc2.ncsu.edu/conferences/nsmc/ (cit. on p. 310).

[401] S. Wu and M. T. Chu. "Markov chains with memory, tensor formulation, and the dynamics of power iteration". In: *Applied Mathematics and Computation* 303 (2017), pp. 226–239. DOI: 10.1016/j.amc.2017.01.030 (cit. on p. 310).

[402] D. Aharonov, A. Ambainis, J. Kempe, and U. V. Vazirani. "Quantum walks on graphs". In: *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*. 2001, pp. 50–59. DOI: 10.1145/380752.380758 (cit. on p. 312).

[403] M. A. Rodriguez and J. H. Watkins. "Quantum Walks with Gremlin". In: *CoRR* abs/1511.06278 (2015). arXiv: 1511.06278 (cit. on p. 312).

[404] K. Xu, J. Chen, H. Wang, and Y. Yu. "Hybrid Graph based Keyword Query Interpretation on RDF". In: *Proceedings of the ISWC 2010 Posters & Demonstrations Track: Collected Abstracts, Shanghai, China, November 9, 2010*. Ed. by A. Polleres and H. Chen. Vol. 658. CEUR Workshop Proceedings. CEUR-WS.org, 2010. URL: http://ceur-ws.org/Vol-658/paper520.pdf (cit. on p. 315).

[405] H. Tong, C. Faloutsos, and J. Pan. "Fast Random Walk with Restart and Its Applications". In: *Proceedings of the 6th IEEE International Conference on Data Mining (ICDM 2006), 18-22 December 2006, Hong Kong, China*. 2006, pp. 613–622. DOI: 10.1109/ICDM.2006.70 (cit. on p. 316).

[406] J. L. Devezas and S. Nunes. "Index-Based Semantic Tagging for Efficient Query Interpretation". In: *Experimental IR Meets Multilinguality, Multimodality, and Interaction - 7th International Conference of the CLEF Association, CLEF 2016, Évora, Portugal, September 5-8, 2016, Proceedings*. Ed. by N. Fuhr, P. Quaresma, T. Gonçalves, B. Larsen, K. Balog, C. Macdonald, L. Cappellato, and N. Ferro. Vol. 9822. Lecture Notes in Computer Science. Springer, 2016, pp. 208–213. DOI: 10.1007/978-3-319-44564-9_17 (cit. on pp. 317, 337).

[407] S. Fortunato, M. Boguñá, A. Flammini, and F. Menczer. "Approximating PageRank from In-Degree". In: *Algorithms and Models for the Web-Graph, Fourth International Workshop, WAW 2006, Banff, Canada, November 30 - December 1, 2006. Revised Papers*. 2006, pp. 59–71. DOI: 10.1007/978-3-540-78808-9_6 (cit. on p. 326).

[408] M. Najork. "Comparing the effectiveness of hits and salsa". In: *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management, CIKM 2007, Lisbon, Portugal, November 6-10, 2007*. 2007, pp. 157–164. DOI: 10.1145/1321440.1321465 (cit. on p. 329).

[409] A. Tonon, G. Demartini, and P. Cudré-Mauroux. "Combining inverted indices and structured search for ad-hoc object retrieval". In: *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '12, Portland, OR, USA, August 12-16, 2012*. 2012, pp. 125–134. DOI: 10.1145/2348283.2348304 (cit. on p. 331).

[410] J. S. Kandola, J. Shawe-Taylor, and N. Cristianini. "Learning Semantic Similarity". In: *Advances in Neural Information Processing Systems 15 [Neural Information Processing Systems, NIPS 2002, December 9-14, 2002, Vancouver, British Columbia, Canada]*. 2002, pp. 657–664. URL: http://papers.nips.cc/paper/2316-learning-semantic-similarity (cit. on p. 333).

[411] J. Devezas and S. Nunes. "Characterizing the hypergraph-of-entity and the structural impact of its extensions". In: *Applied Network Science - Special Issue of the 8th International Conference on Complex Networks and Their Applications* 5.1 (2020), p. 79. ISSN: 2364-8228. DOI: 10.1007/s41109-020-00320-z (cit. on p. 336).

[412] J. L. Devezas and S. Nunes. "Army ANT: A Workbench for Innovation in Entity-Oriented Search". In: *Advances in Information Retrieval - 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14-17, 2020, Proceedings, Part II*. Ed. by J. M. Jose, E. Yilmaz, J. Magalhães, P. Castells, N. Ferro, M. J. Silva, and F. Martins. Vol. 12036. Lecture Notes in Computer Science. Springer, 2020, pp. 449–453. DOI: 10.1007/978-3-030-45442-5_56 (cit. on p. 336).

[413] J. Devezas and S. Nunes. "Characterizing the Hypergraph-of-Entity Representation Model". In: *Complex Networks and Their Applications VIII - Volume 2 Proceedings of the Eighth International Conference on Complex Networks and Their Applications COMPLEX NETWORKS 2019, Lisbon, Portugal, December 10-12, 2019*. 2019, pp. 3–14. DOI: 10.1007/978-3-030-36683-4_1 (cit. on p. 336).

[414] J. Devezas and S. Nunes. "Social Media and Information Consumption Diversity". In: *Proceedings of the Second International Workshop on Recent Trends in News Information Retrieval co-located with 40th European Conference on Information Retrieval (ECIR 2018), Grenoble, France, March 26, 2018*. 2018, pp. 18–23. URL: http://ceur-ws.org/Vol-2079/paper5.pdf (cit. on p. 337).

[415] J. Devezas. *Army ANT: A Workbench for Innovation in Entity-Oriented Search - External Option: Scientific Activities – TREC Open Search*. Research rep. Faculty of Engineering, University of Porto, June 2017. URL: https://hdl.handle.net/10216/110181 (cit. on p. 337).

[416] J. Devezas. *Auditing Open Access Repositories - Free Option: Supervised Study – Digital Archives and Libraries*. Research rep. Faculty of Engineering, University of Porto, May 2017. URL: https://hdl.handle.net/10216/104152 (cit. on p. 337).

[417] J. Devezas and S. Nunes. "Information Extraction for Event Ranking". In: *6th Symposium on Languages, Applications and Technologies (SLATE 2017)*. Ed. by R. Queirós, M. Pinto, A. Simões, J. P. Leal, and M. J. Varanda. Vol. 56. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017, 18:1–18:14. ISBN: 978-3-95977-056-9. DOI: 10.4230/OASIcs.SLATE.2017.18 (cit. on p. 337).

[418] T. Devezas, J. Devezas, and S. Nunes. "Exploring a Large News Collection Using Visualization Tools". In: *Proceedings of the First International Workshop on Recent Trends in News Information Retrieval co-located with 38th European Conference on Information Retrieval (ECIR 2016), Padua, Italy, March 20, 2016*. 2016, pp. 48–53. URL: http://ceur-ws.org/Vol-1568/paper9.pdf (cit. on p. 337).

# APPENDIX

# A PAGERANK-BASED APPROACHES: A COMPONENT-AWARE SURVEY

PageRank is a random walk based approach that was introduced in 1997 as Google's ranking algorithm. Since then, it has been studied and extended for multiple other applications. From query-dependent ranking to community detection and image ranking, PageRank has proven to be flexible, robust and useful in diverse contexts. In this survey, we provide an overview on different types of PageRank applications, covering several interpretations, components and calculation approaches that can be tinkered with to manipulate and model PageRank-based approaches. We provide a normalized notation, focusing on delivering out-of-the-box, power-iteration-ready equations for each type of PageRank we cover. In many cases, we have rewritten the original formulas to provide a vectorized alternative. This way, we aim to provide a comprehensive guide to PageRank, that can be directly used by anyone with access to a linear algebra framework.

When it was first introduced in 1997, PageRank [33] was already an elegant algorithm for measuring the importance of a web page. Since then, we have come a long way to the Multilinear PagerRank [114], developed for the "world of tensors", with multidimensional complexity in mind. At its core, present PageRank variations are still quite similar to the original work by the creators of Google, Larry Page and Sergey Brin. However, its components have now been properly dissected and explored by the scientific community. Over time, PageRank was studied, decomposed and reassembled into morphed versions of itself. Different authors focused on different aspects of PageRank, from the type of smoothing that was used, to the introduction of combined similarities or weights, and the harnessing of personalization for different applications. For instance, PageRank was originally a query-independent weight that took advantage of the web graph to improve text-based search, but it has now been used as a query-dependent weighting approach as well [371, 372]. It has even been explored in the recent field of entity-oriented search [62], where it is frequently applied to entity ranking in knowledge graphs, be it for the query-independent ranking of resources [66, 72, 235], for keyword-based search [65, 67], for recommendation [238], or for class ranking [239].

PageRank is now over 20 years old and, despite its age, it's still being studied. Several research opportunities and challenges still exist, not only regarding applications, but also for delving deeper into weakly explored aspects, like the different types of random walks — e.g., lazy random walks and random walks with back step [373]. Furthermore, PageRank applications to hypergraphs are only just being proposed [374, 375] and novel approaches to entity-oriented search, relying on random walks on hypergraphs [269], are also approaching PageRank.

In this work, we survey PageRank variations and applications, starting at the original PageRank, that emerged in 1997, and ending with Multilinear PageRank, proposed in 2015. The goal is to provide a comprehensive guide to aid the reader in understanding the different types of available PageRanks, the components that can be tinkered with (and how), as well as to provide a manual for directly computing several different PageRank variations, based on power iteration. This should save the reader of the effort of vectorizing some of the approaches, and of spending a large amount of time sorting through the different notations used by different authors.

The remainder of this document is organized as follows. In Section A.1, we present an overview of surveys on random walks and PageRank, describing their

structure and identifying their unique strengths. In Section A.2, we provide an insight into different variations of PageRank, based on interpretations, components, and calculation approaches. We group approaches into several categories, including PageRank without hyperlinks (Section A.2.3), Dirichlet PageRank (Sections A.2.4 and A.2.5), topic-sensitive PageRank (Section A.2.6), weighted PageRank (Section A.2.7) and tensor-based PageRank (Section A.2.8). In Section A.3, we propose an organization of PageRank modeling characteristics, providing a summary table, where we identify these characteristics and present several highlights on each PageRank variation. Finally, in Section A.4, we present some final remarks on PageRank and our vision for the future.

## A.1 RANDOM WALKS IN GRAPHS AND PAGERANK

We present a compilation of relevant surveys on random walks in graphs, as well as directly on PageRank. Lovasz [222] covers several aspects of random walks on graphs, starting with basic notations and facts, like the probability of visiting a node at a given time, the transition matrix representing a Markov chain, the rule of the walk (i.e., how to take steps), as well as the concept of a stationary walk. He then covers the main parameters of random walks and their properties, which includes: *access time* or *hitting time* (number of expected hops required to reach a node, from another given node) and *commute time* (the round-trip number of hops between two nodes); *cover time* (expected time/iterations to reach every node); and *mixing rate* (indicator of the number of convergence steps). They also discuss the connection of random walks to the eigenvalue, covering for instance the fact that the largest eigenvalue of a Markov matrix is one. They then cover the electrical connection, focusing on the role of harmonic functions in random walks, and they delve into further detail regarding the mixing rate. They conclude with a section on sampling based on random walks, a topic that is also covered more recently by Leskovec and Faloutsos [223].

Getoor and Diehl [148] compiled a survey on an area they called "link mining", covering several PageRank approaches on their link-based ranking section. Berkhin [230] provides a survey on PageRank computing, covering the PageRank formulation, as well as several acceleration approaches for faster convergence. In particular, they cover extrapolation methods, the adaptive method where the algorithm stops iterating over individual converged components, a block structure approach that computes PageRank per blocks instead of individual nodes, an approach based on the directed acyclic graph structure of the web, where blocks are formed based on strongly connected components (i.e., sets of nodes that can all reach each other, respecting link direction). They also cover acceleration based on spectral and equation solvers, viewing PageRank as a linear system, as well as based on advanced numerical linear algebra methods. Finally, they also include personalization and PageRank variations, including topic-sensitive PageRank, block personalization, scaled personalization, and query-dependent PageRank. And they cover other relevant issues like stopping criteria, and computing infrastructure.

Chung [228] provides a short and straightforward survey on PageRank, that covers its definition, as well as several methods for approximation, including the classical iterative method, but also the push algorithm, the sharp approximate PageRank, and the random walk based approximation. She covers applications and generalizations of PageRank, including local partitioning and the measurement of node importance, as well as applications like disease spreading containment, trust-based ranking, or gene identification. She also mentions other PageRank variations, like the Kronecker PageRank for multi-attribute networks [376] or the heat kernel PageRank [164], both valuable contributions by the survey's author.

Gleich [155] provides perhaps the most complete survey of existing PageRank variations and applications, covering the mathematics and construction of

PageRank (the notation they use is quite clear and minimalist), its applications and generalizations. Some of the PageRank variations that are uniquely covered in this survey include Reverse PageRank, as well as applications like GeneRank, CiteRank, AuthorRank, FactRank, and several others, which are detailed in-depth in the manuscript.

In this survey, we provide a manual or guide to different types of PageRank, with multiple interpretations, by focusing on their components and the many ways the algorithm can be manipulated and altered. In order to facilitate experimentation and usage, we focus on power iteration to compute PageRank, normalizing notation across the literature and, when required, rewriting some of the iterative approaches to fit this computation model. Our goal is to highlight the components that make PageRank, as well as the variations introduced in the literature, in order to facilitate the manipulation and adoption of this pervasive algorithm for custom applications.

## A.2 MORE THAN ONE PAGERANK

PageRank is an extremely flexible node ranking algorithm, that can be used in multiple different applications. From the role of query-independent feature to improve search, to a way of measuring trust in a network of pre-identified trusted and untrusted sources, or even to compute image similarity, its versatility can be attributed to its many interpretations, components, and calculation approaches.

The Markov matrix used to compute PageRank through power iteration can have different interpretations, depending on the underlying context of application and approach:

- **Graph-based** – a graph can be represented by its adjacency matrix $A$, which can then be column-normalized, filling zero-columns with a uniform probability, and obtaining the Markov matrix $\mathcal{M}$ (e.g., this can be applied to the web graph);
- **Similarity-based** – similarities between all combinations of pairs or entities can be represented as a square matrix, which can then be normalized and used as a Markov matrix (e.g., relying on the topic-based similarity between two web pages, instead of using the transition probability based on the hyperlinks);
- **Probabilistic** – at the core of Markov processes, we directly work with probabilities, without the need for an underlying graph or similarity matrix (e.g., considering the prior probability of query terms, the probability of a random surfer jumping to a page given the query term, and the probability of navigating to an adjacent page).

Furthermore, we can also look at PageRank based on its components, be it the Markov matrix and its interpretation, that we previously described, the teleportation vector, or the way both are combined:

- **Markov matrix weights** – the normalized adjacency matrix weights, representing a graph, can be replaced by any probability — or normalized similarity — as long as the matrix remains stochastic (there are, however, obvious computational advantages to keeping the matrix sparse);
- **History** – traditionally, only the previous state (e.g., the preceding web page) is used for computation, however we can consider a larger amount of past states by modeling them as composite states, reducing a higher-order Markov chain to a first-order Markov chain, or by defining a tensor-based PageRank where each added dimension increases the length of the path to be stored (e.g., Multilinear PageRank).

- **Smoothing** – traditionally, PageRank relies on a fixed-coefficient interpolation method (or Jelinek-Mercer smoothing), which is a component that can be easily replaced (e.g., Dirichlet smoothing, weighted average);
- **Personalization** – the original PageRank by Page and Brin [113] relied on a uniform teleport probability, but the authors also proposed personalization, where teleport probability would be based on a prior node importance instead (e.g., applicable to personal search);

An advantage of PageRank, in part due to its multiple contexts of application and interpretations, is the fact that it can also be calculated based on different computation approaches:

- **Iterative** – starting from a uniform probability per node as the initial value of PageRank, iterative updates are done over each node's PageRank, based on its neighbor PageRanks, until convergence (too slow for most applications);
- **Algebraic** – finds the exact solution to PageRank through linear algebra, but it is highly demanding in computational resources, since it must compute the inverse of a dense matrix (rarely or never used in practice);
- **Power iteration** – finds an approximate solution to PageRank by combining the navigation term and the teleportation term into a Markov matrix, which iteratively multiplied over the PageRank vector, starting from a uniform probability (the most frequent approach, reportedly used by Google);
- **Gaussian elimination** – looks at PageRank as a system of linear equations where each variable is the PageRank for a node (used by igraph[1] to significantly improve efficiency, when compared to their prior power iteration implementation);
- **Monte Carlo** – rapidly approximates PageRank based on repeated random samples from the PageRank distribution (i.e., it simulates a series of random walks over the underlying graph, with teleport) — in some cases, a good approximation of PageRank can be determined with only one iteration [154].

In this section, we provide an overview on PageRank variations, covering the previously listed items. We focused on providing ready-to-use formulas, based on power iteration, for each type of PageRank that we cover. This required, in some cases, that the formula was rewritten or vectorized, also modifying the notation so that it was consistent over the whole manuscript. We used verbose, but clear and widely adopted variables, prioritizing graph theory and then information retrieval notation, when possible. Table A.1 provides a commented overview on the symbols that we used in the survey.

### A.2.1 PageRank (the original)

PageRank is modeled after a random web surfer. The idea is that visiting pages in the web can either be done by clicking on a random link from the current page (navigation) or by randomly jumping to a new page (teleportation). Teleportation frequently happens when reaching a sink or when stuck on a cycle in the web graph, meaning no new information is reached and the surfer eventually gets "bored". For applications to the web graph, the probability of navigation is usually defined as 0.85, while the probability of teleportation is usually defined as 0.15. These complementary probabilities are set by a damping factor $d$. The damping factor is used for the Jelinek-Mercer smoothing (interpolation) of the two probability distributions — $d$ for following a link and $(1 - d)$ for jumping to another page. The PageRank

1 See the *old* attribute in: https://igraph.org/r/doc/page_rank.html.

**Table A.1:** Notation used throughout the survey.

| Notation | Description |
|---|---|
| $pr$ | PageRank of a node $i$. In order to distinguish between different PageRank variations, we also use other notation in a similar manner (e.g., $dpr$, $qdpr$). |
| $PR$ | PageRank vector for all nodes. In order to distinguish between different PageRank variations, we also use other notation in a similar manner (e.g., TSPR, FAPR). |
| $d$ | Damping factor. It always represents the probability of navigation (i.e., of following a link as opposed to jumping to a random node). We sometimes also use $\alpha$, $\beta$ and $\gamma$ either when $d$ is not enough (e.g., three weights are used) or when the semantics of $d$ is different despite having a similar application (e.g., sometimes we use $\beta = 1 - d$). |
| $\mathcal{M}$ | Transition matrix without teleport. Usually obtained by normalizing the columns of a graph adjacency matrix, after replacing empty columns by 1. |
| $\widehat{\mathcal{M}}$ | Transition matrix with teleport (or similar factors; e.g. FolkRank considers users, tags and resources, instead of following a hyperlink or jumping to a new page in the web). |
| $A$ | Adjacency matrix of a graph. Binary matrix, with ones in each column position $i$, representing links from node $j$ (columns) to node $i$ (rows). |
| $D$ | Degree matrix. Usually applied as its inverse $D^{-1}$ or its element-wise square root $D^{1/2}$. Since $D$ is a diagonal matrix, the element-wise square-root is equivalent to the square-root of the matrix. However, when using $D^{-1/2}$, only the diagonal is affected, otherwise the remaining elements would result in $\infty$. |
| $J_{n,m}$ | Matrix of ones with $n$ rows and $m$ columns. If only $n$ is provided, the matrix is square. |
| $I_n$ | Identity matrix of size $n$. This matrix is always square and has all ones in the diagonal and zeros in the remaining positions. |
| $\lvert V\rvert$ | Number of vertices. It corresponds to the number of nodes in the input graph. In some cases, there is no underlying graph, but there is still an $\mathcal{M}$ matrix (usually square) that can be represented as a graph and whose size corresponds to $\lvert V\rvert$. |
| $deg^+(j)$ | Outdegree of node $j$. The number of outgoing nodes/edges departing from $j$. |
| $N^+(j)$ | Outgoing neighbors departing from $j$. |
| $N^-(j)$ | Incoming neighbors arriving at $j$. |

equation that the community is familiar with is only presented in the 1998 article describing Google [151]. In this article, there is a known error where $(1 - d)$ should have been multiplied by the uniform probability $\frac{1}{|V|}$ of jumping to any page in the web graph. Here, $|V|$ represents the number of vertices in the unweighted directed web graph $G = (V, E)$, as the number of documents (pages) in the collection (the web). The corrected version is shown in Equation A.1, highlighting the missing factor in bold and normalizing for a consistent notation across the survey. The incoming neighbors of a vertex $i$ are given by $N^-(i)$ and the outdegree of a vertex $j$ is given by $deg^+(j)$, where the minus sign is used for incoming relations and the plus sign is used for outgoing relations.

$$pr(i) = \frac{(1-d)}{\mathbf{|V|}} + d \sum_{j \in N^-(i)} \frac{pr(j)}{deg^+(j)} \tag{A.1}$$

Equation A.1 can also be vectorized, as shown in Equation A.2, in order to directly compute the PageRank vector $PR$ for all vertices. The column vector of ones with $|V|$ columns is represented by $J_{|V|,1}$, the square matrix of ones of size $|V|$ is represented by $J_{|V|}$ and $\mathcal{M}$ is the Markov matrix representing the column-normalized adjacency matrix $A$ of graph $G$, assuming that columns model outgoing edges —

this is not always the convention, but it is useful in order to avoid the needless use of transposition, for instance in Markov matrices.

$$PR = \frac{1-d}{|V|} \cdot J_{|V|,1} + d\, \mathcal{M} \cdot PR \tag{A.2}$$

Equation A.2 can also be rewritten as Equation A.3, where $\widehat{\mathcal{M}}$ is also a Markov matrix, making PageRank solvable through power iteration — starting with a random probability vector PR (i.e., where L1-norm must be 1), repeat the first part of Equation A.3 until convergence, usually measured through the L2-norm of the difference between the PR vector at iterations $t+1$ and $t$. Notice that in Equation A.3 we effectively multiply the teleportation term of PageRank by PR, which we did not in Equation A.2. However, both equations are equivalent simply because, when multiplied by a probability vector, a matrix of ones $J_{|V|}$ is a vector of ones $J_{|V|,1}$.

$$\begin{aligned} PR &= \widehat{\mathcal{M}} \cdot PR \\ \widehat{\mathcal{M}} &= \frac{1-d}{|V|} \cdot J_{|V|} + d\mathcal{M} \end{aligned} \tag{A.3}$$

Curiously, the main article about PageRank [113] presents a slightly different version, shown in Equation A.4, where instead of Jelinek-Mercer smoothing, a multiplication by a variable $d$ with different semantics is used — we replace $d$ by $\beta$, since it acts more as the factor equivalent to the complement of $d$ in Equation A.1. Here $\beta$ is still in the interval $[0, 1]$, however it is dynamically computed at each iteration based on the difference in L1-norm between the current and next PageRank vectors, at this point without considering teleportation. Then, while still at the current iteration, $\beta$ is multiplied by $E$, a uniform probability vector of $\frac{1}{|V|}$, and added to the partial PageRank that had been computed based only on the probabilities of following links. The algorithm is usually initialized with a random probability vector $P$ and stops when convergence is achieved by minimizing $\delta$ based on a pre-established $\epsilon$ (e.g., 0.001). The same convergence-control strategy is also used in power iteration, described in Equation A.3, based on the L2-norm instead of the L1-norm.

$$\begin{aligned} PR_0 &\leftarrow P \\ &\textit{loop}: \\ PR_{t+1} &\leftarrow \mathcal{M} \cdot PR_t \\ \beta &\leftarrow \|PR_t\|_1 - \|PR_{t+1}\|_1 \\ PR_{t+1} &\leftarrow PR_{t+1} + \beta E \\ \delta &\leftarrow \|PR_{t+1} - PR_t\|_1 \\ &\textit{while } \delta > \epsilon \end{aligned} \tag{A.4}$$

In their 1998 technical report [113], Page et al. also explored the various applications of PageRank:

- Web search — for improving results of underspecified queries;
- Traffic estimation — where important web pages, that receive links anywhere from multiple low importance web pages to a small number of important web pages, are more likely to drive traffic;
- Backlink prediction — for instance for priority crawling in order of importance;
- User navigation — for informing about page importance prior to clicking;
- Competitor analysis — for identifying the top backlinks of competing web sites and any missing link potential.

### A.2.2 Personalized PageRank

The concept of personalization was also introduced by Page et al. [113], simply by assigning a non-uniform teleportation probability to each vertex. For Equation A.1 this means replacing $\frac{1}{|V|}$ with a personalized weight for page $i$ that depends on the particular user's preferences (e.g., the visitation frequency of the user). For Equation A.4, this means replacing vector $E$ with a non-uniform vector (the personalization vector), which can be a single-page $E$, where all personalized weights except one are zero, or a root-level $E$ which only allows teleportation to web site main pages. While personalization in PageRank is frequently achieved through the manipulation of node weights in the teleportation term, there have also been applications based on the manipulation of edge weights in the navigation term [377, 378].

Personalization is a particularly useful characteristic of PageRank and, as such, personalized PageRank has been researched for diverse applications. Avrachenkov et al. [379] proposed Monte Carlo methods for the fast detection of top-k personalized PageRank lists for named entity disambiguation. Pathak et al. [380] proposed an indexing strategy based on the precomputation of personalized PageRank vectors for a selection of hub nodes, in order to provide real-time personalized PageRank computations. Iván and Grolmusz [381] applied it to the study of protein-protein interaction networks. Avron and Horesh [382] used it for community detection, while proposing a time-dependent generalization that could also be configured to compute the heat kernel diffusion vector [164]. Lahoti et al. [383] used the personalized PageRank over an endorsement graph, built using crowd-sourced Twitter lists, to find experts for a given topic query — topics were expressed as labels of the endorsement graph. Musto et al. [238] used personalized PageRank for semantically-aware recommendations based on a tripartite user-item-entity graph — they mapped items into DBpedia entities and extended the graph based on triples associated with the item entity. Anil et al. [384] used personalized PageRank to predict terrorist attacks based on a heterogeneous network — used to model multiple dimensions, such as location, time, target type or organization, while assigning each of them a prior importance.

So far we have shown that PageRank is an elegant metric, that can be modeled as a Markov process, solved using different approaches, and even personalized with little effort for diverse applications. Next, we move further away from the original PageRank and explore other innovations based on this metric.

### A.2.3 PageRank without hyperlinks

PageRank is, at its genesis, a graph-based metric, but it also a probabilistic metric. As such, it can be used despite the existence of an explicit graph, as long as there are underlying dependencies to be modeled. Kurland and Lee [261] applied the idea of PageRank to the reranking of documents — obtained through an initial retrieval algorithm, for a given query. They established links to the nearest-neighbor documents, based on the language models obtained from each document. Specifically, they selected the top generator documents according to the Kullback-Leibler divergence between the language models of the source and target documents. They then proposed and evaluated several centrality metrics for structural reranking, with and without the recursive property of PageRank, and based on the unweighted and weighted versions of the generation graph. They found that structural reranking represented an improvement over the initial ranking and that the weighted graph based on Kullback-Leibler divergences for the top similar documents was a superior option to the unweighted graph.

Jing and Baluja [97] have also proposed VisualRank, a PageRank-inspired approach for image search. As opposed to using hyperlinks, VisualRank is instead based on the similarity matrix computed over all images, retrieved for a given key-

word query. Similarity is calculated based on the number of shared local features over the average number of interest points. The result is a Markov matrix that can be used to compute PageRank for instance through the power method.

### A.2.4 Dirichlet PageRank (PageRank with hub bias)

The original PageRank uses Jelinek-Mercer smoothing to combine the probability distribution of following a link with the probability distribution of randomly jumping to another page. Wang et al. [158, 385] explored another type of smoothing, called Dirichlet smoothing, as an alternative, thus proposing a Dirichlet PageRank (or DirichletRank). Their idea was that the outgoing links of hubs should have a higher probability of being followed and, as such, the probability of randomly jumping to a page should not be uniform, but dependent on the number of outgoing links — an indicator of a good hub. Another way to look at it is that the random walk should be biased towards visiting authorities (i.e., with strong links from hubs), as well as more likely to restart from hubs, since this more closely resembles the expected surfer behavior — it's more probable for the surfer to restart from a good source node to increase the available paths to take.

The authors also identified a zero-one gap problem with the original PagerRank that can be solved in an elegant manner using Dirichlet smoothing. In PageRank, when we reach a sink, the probability of randomly jumping to another page becomes one, but, as soon as there is a single outgoing link, this probability immediately drops to $\beta = (1 - d)$, the complement of the damping factor[1], usually set to $\beta = 0.15$ (or $d = 0.85$).

Equation A.5 shows the formula for the DirichletRank[2] using Dirichlet smoothing based on the number of outgoing links of a page — the idea was inspired by smoothing in language models, where the length of the document is used in a similar manner. Consider $\omega$ as the vector of smoothing factors, $\mathtt{diag}(\omega)$ as the function that generates the diagonal matrix from vector $\omega$, $J_{|V|}$ the square matrix of ones with size $|V|$, and $I_{|V|}$ the identity matrix of size $|V|$.

$$\begin{aligned} DR &= \widehat{\mathcal{M}} \cdot DR \\ \widehat{\mathcal{M}} &= \left[ \mathtt{diag}(\omega) \cdot \frac{1}{|V|} J_{|V|} + \left( I_{|V|} - \mathtt{diag}(\omega) \right) \cdot \mathcal{M}^T \right]^T \\ \omega &= \left[ \omega_1, \ldots, \omega_j, \ldots, \omega_{|V|} \right] \\ \omega_j &= \frac{\mu}{\deg^+(j) + \mu} \end{aligned} \tag{A.5}$$

The applied Dirichlet smoothing can be compared to Jelinek-Mercer smoothing if $\beta$ were to be variable and dependent on the outdegree of an incoming neighbor $j$. This would essentially mean that $\beta_j = \omega_j$ and our damping factor — the complement of $\beta$ — is dynamic instead of static. The $\mu$ parameter in $\omega_j$ can, for example, be set to the mean number of outgoing links (i.e., the average outdegree), thus assigning a $\omega_j = 0.5$ weight to nodes with an average outdegree, and a weight of one to sink nodes (which always results in teleportation), with the weight for $\deg^+(j) > \mu$ resulting in $\omega_j < 0.5$ and vice-versa (see Zhai and Lafferty [386] for an example in information retrieval based on document length $|d|$ instead of $\deg^+(j)$).

1 The authors use $\lambda$ to refer to the complement of the damping factor, however we opted to use $\beta$, which is more commonly used in language models from information retrieval.

2 Notice that calculations were made more explicit (e.g., using the identity matrix $I_{|V|}$), and matrices were transposed to match our column-stochastic $\mathcal{M}$ and reordered for consistency with previous PageRank equations in the survey.

### A.2.5 Dirichlet PageRank (for trust-based ranking)

Another type of Dirichlet PageRank, now based on a Dirichlet boundary condition and unrelated to Dirichlet smoothing, has also been proposed as a necessary tool for ranking based on trust and distrust [387, 388]. Chung et al. proposed that a ranking algorithm that took into account a subset S of nodes in a graph was required for the development of trust-based ranking algorithms. They also proposed several applications of Dirichlet PageRank, including the computation of PageRank over a graph of negative links or the computation of PageRank taking into account a set of trusted friends in a social network.

$$\begin{aligned} \mathrm{dpr}(v) &= \begin{cases} \beta\ e_v + (1-\beta) \sum_{u \in V} \mathrm{dpr}(u) W_{u,v} & \text{if } v \in S \\ \sigma_v & \text{otherwise} \end{cases} \\ W &= \frac{1}{2}\left(I_{|V|} + D^{-1} \cdot A\right) \end{aligned} \tag{A.6}$$

Equation A.6 shows how to calculate the Dirichlet PageRank. Consider $\beta = 1 - d$, the teleport probability vector $e$, with $e_v$ being the respective teleport probability for vertex $v$ (e.g., $\frac{1}{|V|}$, if $e$ is uniform), the lazy transition probability matrix $W$, computed based on the identity matrix $I_{|V|}$ of size $|V|$, the inverse of the degree matrix $D^{-1}$ and the binary adjacency matrix $A$ for an undirected graph, and the boundary conditions vector $\sigma$, usually quantifying trust (positive value) or distrust (negative value).

### A.2.6 Topic-sensitive PageRank

Haveliwala [156, 389] proposed a contextual PageRank, based on a predefined set of topics — they used the 16 top-level of the Open Directory Project[1] — where context is given by the terms in pages containing the query terms. A PageRank score is precomputed for each topic $j$, based on a personalized probability vector $E_j$ for teleportation — each column $j$ of $E$ is a uniform probability vector for a topic. For each vector $E_j$, the author uses $\frac{1}{|T_j|}$ according to the number of pages $|T_j|$ about topic $j$, instead of a fixed $\frac{1}{|V|}$ probability. When issued, queries are classified, obtaining a distribution $w$ over the available topics — this determines the contribution of each topic to the query, which is then multiplied by the teleportation probability matrix $E$ and used for personalization.

$$\begin{aligned} \mathrm{TSPR} &= \widehat{\mathcal{M}} \cdot \mathrm{TSPR} \\ \widehat{\mathcal{M}} &= \beta\ \mathrm{diag}(E \cdot w) \cdot J_{|V|} + (1-\beta)\ \mathcal{M} \end{aligned} \tag{A.7}$$

As illustrated in Equation A.7, the topic-sensitive PageRank can be calculated using the original PageRank equation, if the weighted teleportation probability vector $E \cdot w$ is provided as the personalization vector — the result is a simple weighted average of the teleportation probability matrix $E$, according to the topic distribution, probability vector $w$, over the query.

There has been more work on the topic-sensitive PageRank, namely regarding its usefulness, improvements for better query sensitivity and the combination with content similarity metrics. Al-Saffar and Heileman [390] compared the top ranking pages for several combinations and variations of the global PageRank (the original), the topic-sensitive PageRank and the personalized PageRank. The goal was to determine whether these metrics were independently useful for varying levels of personalization. While they found some overlap between the top ranked pages, indicating that the global PageRank might be used as a starting point to compute the

1 http://odp.org

personalized or topic-sensitive versions, the fraction of common pages was still low, particularly for higher personalization levels, thus justifying the usefulness of the different metrics. Hatakenaka and Miura [371] proposed a topic-driven PageRank, as a better query-sensitive alternative to topic-sensitive PageRank. Their approach supports multiple topics per document and is applicable to documents without links through the creation of a graph based on content and topic similarity. They also normalized PageRank taking into account the possibly unbalanced number of documents per topic. Document ranking, for a given topic and query, can then calculated as described in Equation A.8.

$$\text{score}(d, q) = \sum_{j=1}^{n} T_{jd} \cdot PR'_{jd} \cdot Q_{jq} \tag{A.8}$$

In this equation, $T_{jd}$ represents the topic similarity between topic $j$ and document $d$, $PR'_{jd}$ represents the "balanced" PageRank for topic $j$ and document $d$, and $Q_{jq}$ represents the similarity between topic $j$ and query $q$ based on the word distribution of all documents in topic $j$. The major difference between topic-driven and topic-sensitive PageRank is that now the probability vector for a topic is not uniform, and both topics and queries must surpass a given threshold of similarity with the topic, otherwise they won't contribute to the score — this excludes less relevant documents and reduces topic drift. Rezvani and Hashemi [391] combined the topic-sensitive PageRank with content similarity metrics like the cosine similarity and the Jaccard index, experimenting with different query classifiers — naive Bayes and maximum entropy. Equation A.9 illustrates the main idea behind their proposal, showing how they calculate PageRank for a given topic $T$, represented by its documents, considering the similarity $\text{Sim}_T(\cdot,\cdot)$ between pages from topic $T$. Consider $N^-(\cdot)$ and $N^+(\cdot)$ as the set of incoming and outgoing pages, respectively.

$$\begin{aligned} \text{SimPR}_T(i) = {} & \frac{1-d}{|T|} \\ & + d \sum_{j \in N^-(i)} \text{SimPR}_T(j) \, \frac{\text{Sim}_T(j,i)}{\sum_{k \in N^+(j)} \text{Sim}_T(j,k)} \end{aligned} \tag{A.9}$$

Notice that the PageRank of incoming pages is now distributed according to the normalized similarity over all their outgoing links instead of using the outdegree as is customary. They obtained the best results for the topic-sensitive PageRank when using the cosine similarity metric and naive Bayes to classify queries. Richardson and Domingos [372] have also proposed a query-dependent PageRank based on the directed (or intelligent) surfer model, where a surfer is more likely to follow a link that leads to a page related to the issued query. Equation A.10 illustrates the calculation of the query-dependent PageRank for a query $Q$ with multiple terms $q$.

$$\begin{aligned} \text{qdpr}_Q(i) &= \sum_{q \in Q} P(q) \; \text{qdpr}_q(j) \\ \text{qdpr}_q(i) &= (1-d) \; P'_q(i) + d \sum_{j \in N^-(i)} \text{qdpr}_q(j) \; P_q(j \to i) \end{aligned} \tag{A.10}$$

$P(q)$ represents the prior probability of a query term — i.e., the probability that a term $q$ appears in queries. $P'_q(i)$ represents the probability of a surfer randomly jumping to a page $i$ given the query term $q$ — this can be for instance calculated as the TF-IDF for page $i$ and query $q$, normalized by the sum of all TF-IDF for all pages. $P_q(j \to i)$ represents the transition probability from page $j$ (one of the incoming neighbors $N^-(i)$ of $i$) to page $i$, based on the relevance to the query — this can be for instance calculated as the TF-IDF for page $i$ and query $q$, normalized over all TF-IDF of outgoing neighbors of $j$. This represents a similar, albeit more

general approach, when compared to the one proposed by Rezvani and Hashemi in Equation A.9.

Finally, another case of topic-sensitive PageRank, called FolkRank, was proposed by Hotho et al. [392], over a 3-uniform undirected hypergraph for users, tags and resources, representing a folksonomy. The hypergraph was converted into a tripartite weighed undirected graph and PageRank was adapted to be computed over this graph, taking into account its weights and the principle that "a resource which is tagged with important tags by important users becomes important itself". Equation A.11 shows the calculation of the folksonomy-adapted PageRank for a transition matrix $\mathcal{M}$ and a uniform personalization vector $e$. As opposed to the original PageRank, there are three weights that sum to one — $\alpha$ regulates the speed of convergence, $\beta$ controls the probability of randomly following a link and $\gamma$ controls the probability of jumping to another vertex.

$$\text{FAPR} = \alpha\text{FAPR} + \beta\mathcal{M}^{T} \cdot \text{FAPR} + \gamma e \tag{A.11}$$

Building on the folksonomy-adapted PageRank (FAPR), the authors proposed FolkRank, where the personalization vector $e$ of user preferences over other users, certain tags or particular resources was used to determine the topic. Then, an $\text{FAPR}_0$ was calculated for $\beta = 1$ (i.e., solely based on tripartite user-tag-resource relations) and an $\text{FAPR}_1$ was calculated for $\beta < 1$ (i.e., also considering user preferences). The final score assigned to a vertex was based on the difference between the two scores $\text{FAPR}_1 - \text{FAPR}_0$.

### A.2.7 Weighted PageRank

Dimitrov et al. [157] studied the network of transitions in Wikipedia, based on the actual number of visits from page $i$ to page $j$ — obtained through the merge of a link graph, built from a Wikipedia dump[1], and data from the Wikipedia Clickstream[2]. Their goal was to understand which features made a link successful (i.e., generate a high number of transitions). With that in mind, they studied network features (degree, PageRank, coreness[3]), semantic similarity features (textual and topical similarities) and visual features (area of page containing the link, and screen display coordinates of the link). Using descriptive statistics on boolean features (e.g., *target degree < source degree*) and a mixed-effects hurdle model, they found that: (1) there is a preference to move towards lower degree pages (i.e., the periphery of the network); (2) navigation to semantically similar articles is more prominent; and (3) there are more clicks on the top and left of the page, while navbox links, at the bottom of the page, are infrequently clicked. Through the integration of these features into Markov models they were able to propose a weighted PageRank where the Markov matrix represented a normalized matrix for a given feature or combination of features, generally describing the target vertex (e.g., $\frac{1}{\sqrt{\text{coreness}_j}}$, for a target vertex $j$). Equation A.12 shows how the weighted PageRank is computed, considering a features matrix with elements $f_{i,j}$ that, after normalization, assumes the role of a Markov matrix — notice the resemblence with Equation A.9, where the similarity $\text{Sim}_T(\cdot,\cdot)$ could be seen as yet another feature.

$$wpr(i) = \frac{1-d}{|V|} + d \sum_{j \in N^-(i)} wpr(j) \frac{f_{j,i}}{\sum_{k \in N^+(j)} f_{j,k}} \tag{A.12}$$

1 https://archive.org/details/enwiki-20150304
2 https://figshare.com/articles/Wikipedia_Clickstream/1305770
3 The authors use the k-core designation to refer to the coreness of a vertex $v$, that is, the maximum $k$ over all k-cores $v$ belongs to.

Their best results for the weighted PageRank were obtained based on the coreness and visual features — the Spearman rank correlation to the indegree (sum of incoming transitions) was highest for these features.

### A.2.8 Multilinear PageRank

Gleich et al. [114] have generalized PageRank to higher-order Markov chains, using transition tensors to model historical information. For example, for a second-order Markov chain, a transition tensor of three dimensions would model the probability of visiting a vertex, given the previous two visited vertices. In order to compute their proposed multilinear PageRank, they took advantage of tensor flattening and the Kronecker product of the PageRank vector. Tensor flattening was used to convert a tensor of arbitrary dimension to an analogous matrix. Flattening can happen over any dimension of the tensor, but for multilinear PageRank it was done over the first dimension — for example, an $n \times n \times n$ tensor would result in an $n \times n^2$ matrix $\mathcal{R}$ of the horizontal concatenation of each of the $n$ matrices that formed the third dimension of the tensor. This can also be seen as the process of reducing the higher-order Markov chain to a first-order Markov chain, thus ensuring it still respects the Markov property. The Markov property says that a future state can only depend on the present state (i.e., it is memoryless) — $\mathcal{R}$ can be seen as a matrix where rows are target vertices and columns are source sequences of vertices, each forming a composite, but single state. After calculating $\mathcal{R}$, the Kronecker product was used to generate an extended PageRank vector, which was then multiplied by the flattened tensor — the result of the multiplication thus generates a vector of size $|V|$, while considering additional historical information. Equation A.13 shows the calculation of the multilinear PageRank and provides an intuition as to compute it using power iteration — it also serves to illustrate how we can generalize the power iteration to tensors; here, for $m = 2$ dimensions, we fallback to the traditional transition matrix (i.e., $\mathcal{R} = \mathcal{M}$) with only be one term in the Kronecker product, resulting in the same equation as the original PageRank (Equation A.2).

$$\text{MLPR} = \frac{1-d}{|V|} J_{|V|,1} + d\,\mathcal{R} \cdot (\underbrace{\text{MLPR} \otimes \ldots \otimes \text{MLPR}}_{m-1 \text{ terms}}) \tag{A.13}$$

The authors also delved into other alternatives, besides power iteration, for approximating PageRank and, more importantly, they warned that a unique solution for an order $m$ PageRank only exists for $d < \frac{1}{m-1}$, which isn't a restriction of the original, first-order PageRank. The condition for uniqueness was then further studied by Li et al. [393], showing that a sharper bound exists than the one proposed by Gleich et al. [114] — i.e., a more relaxed condition can be used. Meini and Poloni [394] further analyzed multilinear PageRank, improving the theoretical understanding of this equation and its solutions, and proposing a numerical approach that was able to solve multilinear PageRank for $d \approx 1$.

Other loosely related work that deserves a mention includes vertex-reinforced random walks [395] and the spacey random walk [396], which are at the base of multilinear PageRank, as well as tensor spectral clustering [397], for partitioning higher-order network structures such as triangles, and tensor factorization, for predicting user trails [398], both modeled as higher-order Markov chains. For more details on useful mathematical tools, we also recommend Golub and Van Loan [399] for general matrix computations, Manning et al. [77, Ch.8] for understanding low-rank matrix approximations, Dayar [400] for learning about the relation between Markov chains and the Kronecker product, and Wu and Chu [401] for more information on higher-order Markov chains, tensors and the power iteration.

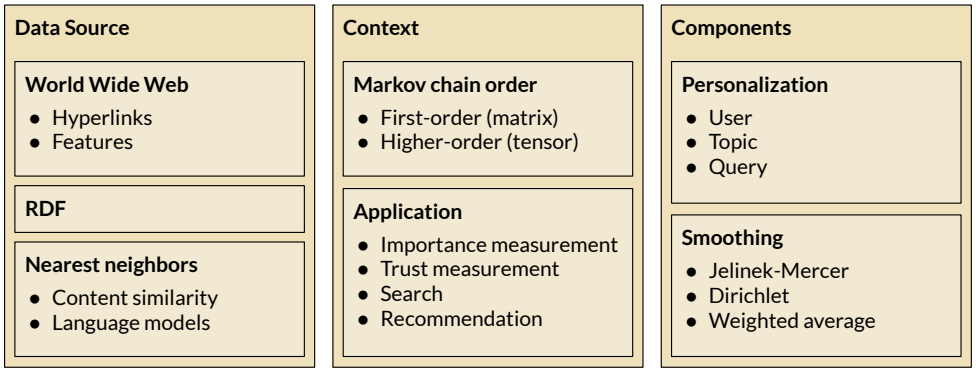



**Figure A.1:** PageRank modeling characteristics.

## A.3 DISCUSSION

We have provided an overview on PageRank algorithms, while attempting to use a consistent notation for an easier reference. We started from the original PageRank, that surfaced in 1997, reaching the Multilinear PageRank in 2015, a generalization for higher-order dependencies represented as tensors. Based on the surveyed PageRank variants, we also propose, as shown in Figure A.1, several modeling characteristics to help organize the different applications and approaches to PageRank. In particular, we considered the data source, the context (Markov chain order and application), and the components (type of personalization and smoothing approach). In regard to the data source, we considered the world wide web, RDF, and nearest neighbor approaches based on content similarity or language models. We considered the context, identifying the Markov chain order, which we classified either as first-order or higher-order[1]. We also identify several applications, considering the measurement of importance (same as the classical PageRank), the measurement of trust, and direct applications for search or recommendation. Regarding components, we identified personalization, considering three types: user-based, topic-based and query-based. We also identified smoothing, considering simple linear combinations based on Jelinek-Mercer and the weighted average, as well as non-linear combinations based on Dirichlet smoothing. In Table A.2, we provide an overview of the surveyed PageRank variants, identifying relevant modeling characteristics. PageRank can be used to rank nodes on a web graph or images on a similarity graph. It can be personalized according to the topics of documents and the issued queries, but it can also be personalized based on user interest. PageRank can be weighted or take into account higher-order dependencies, but it always represents a stationary distribution of a Markov chain, where importance is redistributed until the system becomes stable and further redistribution does not change the outcome.

## A.4 FINAL REMARKS

We have presented an overview on several types of PageRank in an organized manner. We focused on providing utility for an easy selection of which variation of PageRank to use for a particular problem, as well as on identifying the components that can be replaced or modified for custom applications of PageRank. We also

1 Higher-order dependencies are able to account for historical evidence (e.g., paths of length $\ell > 1$), apparently violating the Markov property (i.e., present states depends only on past states). However, past states can be represented as composed states (e.g., $ab \rightarrow c$ instead of $a \rightarrow b \rightarrow c$) with an associated overall probability. This strategy ensures that the Markov property is not broken.

listed the different approaches to solving PageRank, as well as its multiple interpretations in graph theory, probability theory, or a similarity-based context.

Despite its age already surpassing 20 years, PageRank is still a noteworthy algorithm, not only for its elegance, but also for its flexibility due to its multiple interpretations and component-wise variations. At its core, it models a random walk, where there is both a probability of following a neighboring link, but also a probability of jumping to a random node in the graph. The Markov chain does not require an underlying graph, but this is what make PageRank both graph-based and probabilistic.

## A.5 CHALLENGES AND IDEAS FOR THE FUTURE

Challenges for the future include the continued exploration and development of PageRank algorithms for other types of graphs, such as special cases like bipartite or tripartite graphs, as well as analogous higher-order data structures like hypergraphs. Another opportunity the might arise in time is the exploration of quantum PageRank approaches that would rely on quantum walks [402, 403], instead of random walks, in graphs. A generalization of the previous problem would also include an in-depth study of the impact that the choice of random walk would have on PageRank.

**Table A.2:** Summary of PageRank variations and their modeling characteristics.

| Variation | Modeling characteristics | Highlights |
|---|---|---|
| PageRank [33, 113, 151] | Data source:<br>• World Wide Web → Hyperlinks<br>Context:<br>• Markov chain order → First-order<br>• Application → Search<br>Components:<br>• Personalization → User<br>• Smoothing → Jelinek-Mercer | • The original PageRank first appeared in 1997 [33].<br>• Random surfer model.<br>• Uses teleport as a sort of regularization.<br>• To avoid sinks and cycles.<br>• Widely known formula is in Brin and Page [151].<br>• Original paper was missing the $\frac{1}{\lvert V\rvert}$ probability.<br>• Dynamic computation of $d$ in Page et al. [113].<br>• Describes Google at its prototype stage. |
| PageRank without hyperlinks [261] | Data source:<br>• Nearest Neighbors → Language Models<br>Context:<br>• Markov chain order → First-order<br>• Application → Search (Reranking)<br>Components:<br>• Smoothing → Jelinek-Mercer | • Used for reranking.<br>• Uses language models per document.<br>• To create a graph of nearest-neighbors.<br>• Measured using Kullback-Leibler divergence.<br>• And either weighted according to the posterior.<br>• Or unweighted. |
| VisualRank [97] | Data source:<br>• Nearest Neighbors → Content Similarity<br>Context:<br>• Markov chain order → First-order<br>• Application → Search (Image)<br>Components:<br>• Smoothing → Jelinek-Mercer | • Used for reranking.<br>• Based on results for a keyword query.<br>• Builds a similarity graph.<br>• Based on image features.<br>• Number of shared local features.<br>• Over the average number of interest points. |
| Dirichlet PageRank [158, 385] (Hub Bias) | Data source:<br>• World Wide Web → Hyperlinks<br>Context:<br>• Markov chain order → First-order<br>• Application → Importance Measurement<br>Components:<br>• Smoothing → Dirichlet | • Uses Dirichlet smoothing.<br>• Instead of Jelinek-Mercer.<br>• Based on the outdegree of incoming vertices.<br>• Analogous to document length in language models. |
| Dirichlet PageRank [387, 388] (Trust Based) | Context:<br>• Application → Trust Measurement<br>Components:<br>• Smoothing → Jelinek-Mercer | • Computed over a subset of vertices.<br>• Lazy transition matrix.<br>• Boundary condition vector.<br>• With positive values for trusted vertices.<br>• And negative values for distrusted vertices.<br>• Outside the subset of ranked nodes. |

(Continued on next page)

Table A.2. Graph-based models models for entity-oriented search. (Continued from previous page)

| Variations | Modeling Characteristics | Highlights |
|---|---|---|
| Topic-Sensitive PageRank [156, 389] | Data source:<br>• World Wide Web → Hyperlinks<br>Context:<br>• Markov chain order → First-order<br>• Application → Search<br>Components:<br>• Personalization → Topic; Query<br>• Smoothing → Jelinek-Mercer | • Personalization is uniform per topic.<br>• And weighted by topic distribution over the query. |
| Topic-Driven PageRank [371] | Data source:<br>• World Wide Web → Hyperlinks<br>• Nearest Neighbors → Content Similarity<br>Context:<br>• Application → Search<br>Components:<br>• Personalization → Topic; Query | • Personalization is not uniform per topic.<br>• It is based on the topic distribution over a document.<br>• And weighted by topic distribution over the query.<br>• Normalized based on number of documents per topic. |
| PageRank with content similarity [391] | Data source:<br>• World Wide Web → Hyperlinks<br>• Nearest Neighbors → Content Similarity<br>Context:<br>• Application → Search<br>Components:<br>• Personalization → Topic; Query | • Adaptation of topic-sensitive PageRank.<br>• Built a weighted graph per topic.<br>• Based on content similarity.<br>• Similarity normalized per incoming vertex. |
| Query-dependent PageRank [372] | Data source:<br>• World Wide Web → Hyperlinks<br>• Nearest Neighbors → Content Similarity<br>Context:<br>• Application → Search<br>Components:<br>• Personalization → Topic; Query | • Intelligent/directed surfer model.<br>• Similar to topic-driven PageRank.<br>• And PageRank with content similarity.<br>• Based on the relevance of the query to all pages.<br>• And the relevance of the query to linked pages. |
| FolkRank [392] | Context:<br>• Application → Search<br>Components:<br>• Personalization → User; Topic; Query<br>• Smoothing → Weighted Average | • 3-uniform hypergraph converted to tripartite graph.<br>• Of users, tags and resources (a folksonomy).<br>• Uses weighted average smoothing.<br>• Personalizable over users, tags and/or resources.<br>• Computed as the difference of PageRanks.<br>• For folksonomy versus personalization. |
| Weighted PageRank [157] | Data source:<br>• World Wide Web → Features<br>Context:<br>• Markov chain order → First-order<br>• Application → Importance Measurement<br>Components:<br>• Smoothing → Jelinek-Mercer | • Reasonable surfer model.<br>• Network, semantic and visual features.<br>• Based on target vertex.<br>• Normalized per incoming vertex.<br>• Transition matrix per feature. |
| Multilinear PageRank [114] | Context:<br>• Markov chain order → Higher-order<br>Components:<br>• Smoothing → Jelinek-Mercer | • Spacey random surfer model.<br>• Based on higher-order Markov chains.<br>• Generalization based on tensor flattening.<br>• And Kronecker product. |

# B FATIGUED RANDOM WALKS

## Contents



In this appendix, we introduce the concept of fatigue, which can be used to temporary restrict, reduce or completely block access to individual nodes or edges in a graph, during a traversal. This idea was inspired by neuronal fatigue, as briefly mentioned by Jon von Neumann in his contribution to the Silliman Memorial Lectures on the relations between the computer and the brain [79]. We explore two different applications based on this cognitive analogy. First, we apply cycles of fatigue to nodes and hyperedges visited during the computation of the random walk score. We then study the performance impact in the hypergraph-of-entity general retrieval model. Our goal is to understand whether imitating a behavior that is a part of the brain could improve graph-based retrieval. Secondly, we propose an approximation of fatigue as an extension to PageRank, applied over a graph using power iteration. We evaluate the quality of Fatigued PageRank as a standalone node ranking metric, as well as a query-independent feature for improving classical ranking functions, such as BM25.

The structure of this appendix is organized as follows:

- **Section B.1** presents the idea of bringing neuronal fatigue from the brain into the computer, in particular as an application to graph traversal computations.
- **Section B.2** proposes fatigued random walks in hypergraphs, applying it to the random walk score [§B.2.1]. We evaluate efficiency and effectiveness, measuring the impact of node and hyperedge fatigue [§B.2.2], and we compare the rankings of the baseline model with the best fatigued model, as well as the random walk score with and without fatigue [§B.2.3].
- **Section B.3** introduces a novel version of PageRank. We begin by suggesting a random explorer model to illustrate the impact of fatigue in random walks [§B.3.1], formalizing Fatigued PageRank and illustrating its computation through power iteration as an extension of PageRank [§B.3.2]. We assess the effectiveness of Fatigued PageRank as a node centrality metric, when compared with baselines like the indegree, HITS authority or PageRank [§B.3.3]. We also explore the impact of Fatigued PageRank in information retrieval, when used as a query-independent feature, in combination with BM25, versus other graph-based metrics [§B.3.4].

## B.1 NEURONAL FATIGUE IN COMPUTER SCIENCE

In preparation for Yale's Silliman Memorial Lectures[1], von Neumann highlighted the importance of jointly studying the computer and the brain [79], a work to be published posthumously for the first time in 1958. In fact, with his lecture, he provided enough common ground for crossover work between computer science and neuroscience, bringing the areas closer together. One of the ideas studied in von Neumann's lecture was the fact that a neuron will become fatigued, for a period of time, after having been stimulated.

> *However, this is not the most significant way to define the reaction time of a neuron, when viewed as an active organ in a logical machine. The reason for this is that immediately after the stimulated pulse has become evident, the stimulated neuron has not yet reverted to its original, prestimulation condition. It is fatigued, i.e. it could not immediately accept stimulation by another pulse and respond in the standard way. [...] It should be noted that this recovery from fatigue is a gradual one [...]"*
>
> – John von Neumann, The Computer and the Brain

Despite the impact neuroscience has had in computer science (e.g., neural networks), not many analogies with fatigue have been proposed. In fact, to our knowledge, only Xu and Yu [404] have used fatigue, in the context of neural networks, as a part of a revised version of backpropagation for spam filtering.

In this appendix, we apply fatigue to nodes and hyperedges in the hypergraph-of-entity, treating them as neurons that have been "stimulated" by a traversal during a random walk. We also introduce node fatigue as a signal used in the computation of PageRank. In Fatigued PageRank, node importance is measured by exploring the topology of the graph through random walks, while taking into account the probability of a node getting fatigued — we call this the random explorer model.

## B.2 FATIGUED RANDOM WALKS IN HYPERGRAPHS

Hypergraphs are data structures capable of capturing supra-dyadic relations. We can use them to model binary relations, but also to model groups of entities, as well as the intersections between these groups or the contained subgroups. In Chapters 7, 8 and 9, we explored the usage of hypergraphs as an indexing data structure, in particular one that was capable of seamlessly integrating text, entities and their relations to support entity-oriented search tasks. As more information is added to the hypergraph, however, it not only increases in size, but it also becomes denser, making the task of efficiently ranking nodes or hyperedges more complex. Random walks can effectively capture network structure, without compromising performance, or at least providing a tunable balance between efficiency and effectiveness, within a nondeterministic universe. For a higher effectiveness, a higher number of random walks is usually required, which often results in lower efficiency, as we can see in Table 9.5, in Chapter 9. Inspired by von Neumann and the neuron in the brain, we propose and study the usage of node and hyperedge fatigue as a way to temporarily constrict random walks during keyword-based ad hoc retrieval. We find that we were able to improve search time by a factor of 32, but also worsen MAP by a factor of 8. Moreover, by distinguishing between fatigue in nodes and hyperedges, we are able to find that, for hyperedge ranking tasks, we consistently obtained lower MAP scores when increasing fatigue for nodes. On the other hand, the overall impact of hyperedge fatigue was slightly positive, although it also slightly worsened efficiency.

1 https://en.wikipedia.org/wiki/Silliman_Memorial_Lectures

Initial experiments with the hypergraph-of-entity had led to acceptable indexing times, but high search times. Depending on the collection, a query might take over 10 minutes to run for random walks of length $\ell = 2$ and repeats $r = 10{,}000$ (cf. Table 9.5). Although we could execute a query in approximately 10 seconds for $\ell = 2$ and $r = 100$, at this point the algorithm wouldn't converge, thus outputting dissimilar rankings for repeated runs with the same configuration[1]. This made the ranking function less useful, as it got closer to a random ranking of a random set of documents connected to the seed nodes (i.e., nodes representing the query in the hypergraph-of-entity). In order to make the hypergraph-of-entity useful in practice, we need to improve efficiency for large values of $r$, which correspond to points of high effectiveness and convergence[2].

We adopt the idea of fatigue for random walks, introducing node and hyperedge fatigue as the number of cycles during which a given node or hyperedge is not available for visitation or traversal, as random steps are taken. Our initial hypothesis was that the addition of fatigue to the retrieval model would result in a significant overhead from maintaining and decrementing several variables of fatigue per cycle. On the other hand, regarding effectiveness, we didn't know what to expect — we were led by the idea that, if this was a part of the neuronal process, then it should be relevant for effective cognition and thus might improve our ranking function. Contrary to our initial hypothesis and, as you will see in Section B.2.2, we found that the introduction of node fatigue significantly improved efficiency, however it decreased effectiveness, maintaining the tradeoff that we had evidenced before.

### B.2.1 Introducing fatigue in random walk score

Random walks have been at the core of centrality metrics like PageRank or personalized PageRank [113]. While PageRank is computed for the whole graph, personalized PageRank is computed for a localized area of the graph, based on a set of seed nodes. In both algorithms there is the probability that we follow a random outgoing edge. Given the complementary event that we don't, in PageRank we jump to a random node in the graph, but in personalized PageRank we always jump to one of the seed nodes instead, resulting in a behavior analogous to a random walk with restart [405]. The random walk score ranking function is similar to personalized PageRank, where our personalization is based on a keyword query and we use fixed length random walks, starting from each seed node, that only jumps back to its seed node after $\ell$ steps instead of doing it randomly. Each seed node represents an expansion of a query term to the entities it might refer to in the hypergraph (i.e., we follow all directed links between a query term and its neighboring entities). As such, departing from closer together seed nodes reinforces the weight of the nodes/hyperedges they all cover, while departing from further apart seed nodes leads to a middle-ground, reinforcing the weights of nodes/hyperedges in the intersecting borders instead.

The idea is to first reach an open interpretation of the query and, only then, as a part of the ranking process, close in on the actual sense of the query. By cross-referencing information based on all query terms, we attempt to diminish ambiguity, while at the same time performing ranking. For example, if we search for `[ Eiffel Tower ]`, we will expand to multiple entities mentioning *"Eiffel"* (at least *Gustave Eiffel* and *Eiffel Tower*) and multiple entities mentioning *"Tower"* (e.g., *Eiffel Tower*, *First National Bank Tower*, *The Regal Tower*). It is through the combination of the neighborhoods of expanded-to entities that we can understand that the query is referring to *Eiffel Tower*. While this is a straightforward example, with little am-

1 Kendall's concordance coefficient $W \approx 0.84$ for 100 iterations with $\ell = 2$ and $r = 100$, as opposed to $W \approx 0.99$ with $\ell = 2$ and $r = 10{,}000$.

2 Notice that, when mentioning *high effectiveness*, we are considering previous instances of the hypergraph-based model, instead of establishing a comparison to the state of the art, as we are still far from that target.

biguity, more complex queries can only be segmented probabilistically based on existing knowledge [406]. Take for instance `[ Gustave Eiffel tower construction ]`, where we could, at the very least, consider both *Gustave Eiffel* and *Eiffel tower* as valid (but overlapping) entities, despite only one of the options being meant by the user. On the other hand, the user might not provide enough information for disambiguation and query understanding, in which case the best option for the search engine is to consider the most probable segmentation and interpretation according to the corpus and/or knowledge base.

Using the seed nodes as an expanded representation of the query in the hypergraph, we then launch a random walk of length $\ell$ from each seed node, repeating this process $r$ times. Our expectation is that, given the appropriate hypergraph structure and restrictions (e.g., node/hyperedge weights, fatigue), we should be able to converge at a ranking of nodes/hyperedges based on the visitation frequency, for a sufficiently large $r$. We then collect the nodes and/or hyperedges that represent the target unit(s) of retrieval (documents and/or entities). We call this ranking function the random walk score, $RWS(Q, \mathcal{H}, \ell, r)$, for a given query $Q$ and hypergraph $\mathcal{H} = (V, E)$, where $V$ is the set of all vertices and $E$ is the set of all hyperedges, with $E_j$ either being a set of vertices, for undirected hyperedges, or a tuple with two sets of vertices, for directed hyperedges.

Previous experiments with *RWS*, namely in TREC 2018 Common Core track (Section 9.2.2), have resulted in low or inconsistent effectiveness and worrisome efficiency, where queries frequently required several minutes to run, in order to converge to a stable ranking. The hypergraph-of-entity is a collection-based representation that links all available data, structured and unstructured, for a given corpus. This means that, whenever we query the hypergraph, we have access to a complete body of knowledge, but also that we are potentially required to traverse a high number of paths before converging to a ranking. The question is then how to reduce the number of traversed paths without impacting effectiveness. In this section, we introduce and test fatigue in random walks in order to determine whether it can improve efficiency by acting as a controller for exploration of the untraveled paths. Fatigue as a restriction is not unlike the seed node introduced in personalized PageRank to focus on local exploration.

In order to propose a fatigued extension of random walk score, let us now consider von Neumann's description of fatigue (Section B.1) and a neuron–node / neuron–hyperedge analogy. This tells us that, immediately after a node or hyperedge is traversed, it should enter a state of fatigue, and thus block random walk visitations for a given period of time $\Delta_{nf}$ (node fatigue) or $\Delta_{ef}$ (hyperedge fatigue). Time passes with each random step taken, meaning that, for every visited node $i \in V$ and hyperedge $j \in E$, we must initialize and maintain a hash table, respectively with $\Delta^i_{nf}$ and $\Delta^j_{ef}$ fatigue statuses that are decremented at each step. Each fatigued element is then excluded from the random sampling, when deciding which node or hyperedge to visit next at a random step. An element stops being fatigued when its fatigue status reaches zero, at that point being removed from the appropriate hash table. The addition of fatigue to the retrieval model results in a random walk score $RWS(Q, \mathcal{H}, \ell, r, \Delta_{nf}, \Delta_{ef})$, extended with the number of steps of fatigue for nodes and hyperedges, where $RWS(Q, \mathcal{H}, \ell, r) = RWS(Q, \mathcal{H}, \ell, r, \Delta_{nf} = 0, \Delta_{ef} = 0)$.

### B.2.2 Retrieval performance assessment

Using Lucene, we indexed the text block of the smaller subset of INEX 2009 10T-NL Wikipedia collection, using TF-IDF and BM25 with $k_1 = 1.2$ and $b = 0.75$ as our baselines. We then indexed the text and knowledge blocks (i.e., terms, entities and their relations) using two versions of the hypergraph-of-entity: the base model, and an extension of the base model using *synonym* and *context* hyperedges. For both hypergraph-of-entity models, we used RWS with length $\ell = 2$ and repeats $r = 1{,}000$.

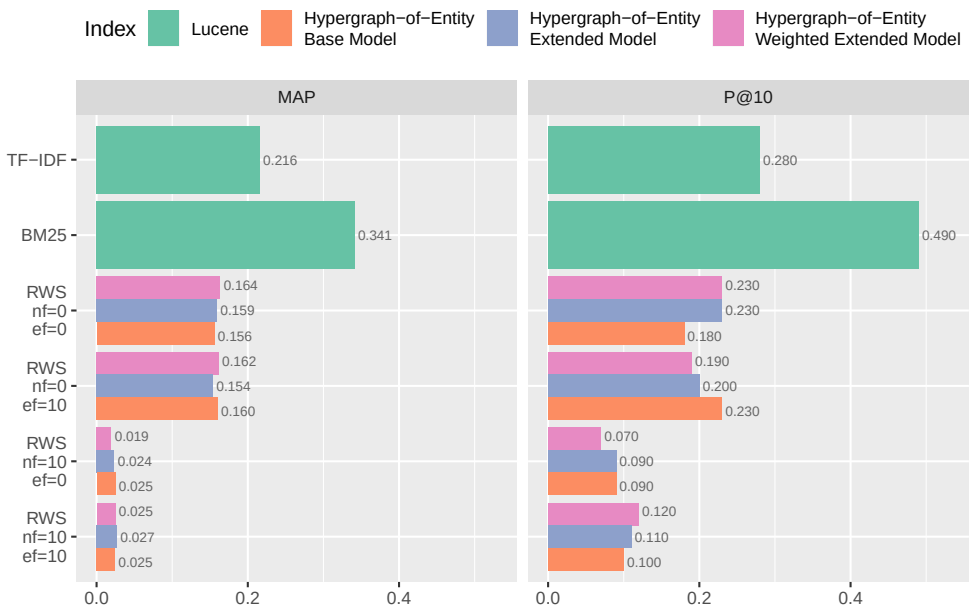



**Figure B.1:** Evaluation metrics for Lucene baselines and hypergraph-of-entity models using different combinations of node and hyperedge fatigue for RWS.

We then experimented with all combinations of node and hyperedge fatigue cycles within $\Delta_{nf} \in \{0, 10\}$ and $\Delta_{ef} \in \{0, 10\}$, resulting in four runs for each hypergraph-of-entity model, in addition to the two baseline runs. Our assessment was based on the Mean Average Precision (MAP), as well as the Precision at a cutoff of $n$ (P@10). We also measured indexing and search times to capture the impact that different models and parameter configurations had in efficiency.

**EFFICIENCY** Indexing with Lucene took 25s 889ms, averaging 3.50ms per document. Indexing with the hypergraph-of-entity took 1m 9s 687ms, averaging 9.42ms per document, for the base model. The times were similar for the extended models, taking 1m 4s 135ms and 1m 17s 193ms, for an average of 8.67ms and 10.43ms per document, respectively for the non-weighted and weighted versions. Regarding search time, Lucene took on average, per query, 451ms for TF-IDF and 269ms for BM25. Hypergraph-of-entity was less efficient during search, taking on average, per query, between 1m 4s 620ms and 5m 31s 286ms for the base and extended models, when $\Delta_{nf} = 0$, that is, either without fatigue or with hyperedge fatigue only. On the other hand, for $\Delta_{nf} = 10$, average query time ranged between 834ms and 1s 49ms, but it also resulted in a lower overall effectiveness.

**EFFECTIVENESS** We experimented with all four different combinations of node and hyperedge fatigue: no fatigue ($\Delta_{nf} = 0$, $\Delta_{ef} = 0$), node fatigue ($\Delta_{nf} = 10$, $\Delta_{ef} = 0$), hyperedge fatigue ($\Delta_{nf} = 0$, $\Delta_{ef} = 10$) and full fatigue ($\Delta_{nf} = 10$, $\Delta_{ef} = 10$). Results are detailed in Table B.1. Figure B.1 shows the MAP and P@10 scores for the hypergraph-of-entity base model, extended model and weighted extended model, when compared to the Lucene TF-IDF and BM25 baselines. A different filling color was used for each index (i.e., for each representation model). As we can see, none of the versions of the hypergraph-of-entity with RWS were able to outperform TF-IDF or BM25. Moreover, node fatigue had a negative effect in performance, while hyperedge fatigue had little to no effect on performance. We also found that the difference in MAP was not statistically significant when comparing TF-IDF and the best runs for RWS (base model with hyperedge fatigue and weighted extended model without fatigue). We obtained p-values of 0.3930 and 0.4813, respectively, for the Mann-Whitney U tests. On the other hand, when com-

**Table B.1:** Performance of random walk score, with different levels of fatigue, over the hypergraph-of-entity.

| Version | | MAP | P@10 | Search Time | |
|---|---|---|---|---|---|
| | | | | Avg./Query | Total |
| Lucene TF-IDF | | 0.2160 | 0.2800 | 451ms | 4s 510ms |
| Lucene BM25 | | 0.3412 | 0.4900 | 269ms | 2s 688ms |
| **Hypergraph-of-Entity Base Model** – $RWS(\ell = 2, r = 1{,}000)$ | | | | | |
| $\Delta_{nf} = 0$ | $\Delta_{ef} = 0$ | 0.1560 | 0.1800 | 1m 14s | 12m 18s |
| $\Delta_{nf} = 0$ | $\Delta_{ef} = 10$ | 0.1601 | 0.2300 | 1m 22s | 13m 39s |
| $\Delta_{nf} = 10$ | $\Delta_{ef} = 0$ | 0.0249 | 0.0900 | 839ms | 8s 387ms |
| $\Delta_{nf} = 10$ | $\Delta_{ef} = 10$ | 0.0246 | 0.1000 | 834ms | 8s 338ms |
| **Hypergraph-of-Entity w/ Syns. & Context** – $RWS(\ell = 2, r = 1{,}000)$ | | | | | |
| $\Delta_{nf} = 0$ | $\Delta_{ef} = 0$ | 0.1594 | 0.2300 | 1m 05s | 10m 48s |
| $\Delta_{nf} = 0$ | $\Delta_{ef} = 10$ | 0.1540 | 0.2000 | 1m 05s | 10m 46s |
| $\Delta_{nf} = 10$ | $\Delta_{ef} = 0$ | 0.0236 | 0.0900 | 955ms | 9s 553ms |
| $\Delta_{nf} = 10$ | $\Delta_{ef} = 10$ | 0.0272 | 0.1100 | 924ms | 9s 242ms |
| **Hypergraph-of-Entity w/ Syns., Context & Weights** – $RWS(\ell = 2, r = 1{,}000)$ | | | | | |
| $\Delta_{nf} = 0$ | $\Delta_{ef} = 0$ | 0.1636 | 0.2300 | 5m 26s | 54m 15s |
| $\Delta_{nf} = 0$ | $\Delta_{ef} = 10$ | 0.1615 | 0.1900 | 5m 31s | 55m 13s |
| $\Delta_{nf} = 10$ | $\Delta_{ef} = 0$ | 0.0195 | 0.0700 | 1s 011ms | 10s 106ms |
| $\Delta_{nf} = 10$ | $\Delta_{ef} = 10$ | 0.0250 | 0.1200 | 1s 049ms | 10s 491ms |

paring the MAP for BM25 and the best run for RWS, we found that the difference was statistically significant, with a p-value of 0.004 in both cases.

As it stands, the introduction of fatigue had little impact, except when considering the P@10 for the base model when using hyperedge fatigue. For that particular case, we were able to increase the performance of the base model without the need for synonyms, context or weights. Despite the small size of the sample, we were, at the very least, able to achieve a similar performance for TF-IDF and RWS, using a hypergraph-based model, instead of an inverted index, and a nondeterministic random walk based approach. The model we propose has the potential to, through its joint representation of text, entities and their relations, unlock novel ranking strategies that take into account all available leads, be it those locked within unstructured text or those explicitly provided through structured knowledge. Beyond document ranking, hypergraph-of-entity can easily support entity ranking, related entity finding and entity list completion.

### B.2.3 Rank correlation analysis

We used Spearman's rank correlation coefficient $\rho$ to compare vectors of positions without ties. Any missing positions were added to either vector, using the lexicographical order for tied documents, based on their ID. Missing documents were assigned synthetic incremental positions, after the last retrieved document, in order to complete the rankings and make them comparable. Given the nondeterministic (but overall converging) character of RWS, we computed and averaged 100 $\rho$ values for the same parameter configuration, in order to obtain a robust insight.

We ran two experiments. The first enabled us to further understand the differences between the baselines and our models. We compared the rankings provided by Lucene TF-IDF, achieving a P@10 of 0.2800, with the rankings provided by RWS, for $\ell = 2$, $r = 1{,}000$, $\Delta_{nf} = 0$ and $\Delta_{ef} = 10$, achieving a P@10 of 0.2300 for the base model (the best with fatigue). In the second experiment, we compared two versions of RWS, with and without fatigue. Particularly, we focused on hyperedge fatigue, since it resulted in a similar performance to the version without fatigue. The second experiment was run over the extended model.

**Table B.2:** Spearman's rank correlation coefficient ρ and Jaccard index J, averaged over 100 repeated retrieval events for the same topic. $\langle \rho_1 \rangle$ and $\langle J_1 \rangle$ compare TF-IDF and the best RWS with fatigue, while $\langle \rho_2 \rangle$ and $\langle J_2 \rangle$ compare RWS with fatigue ($\Delta_{ef} = 10$) and without fatigue. The mean μ and standard deviation σ are shown at the bottom of the table.

| Topic | $\langle \rho_1 \rangle$ | $\langle \rho_2 \rangle$ | $\langle J_1 \rangle$ | $\langle J_2 \rangle$ |
|---|---|---|---|---|
| 2010003 | -0.8400 | 0.9639 | 0.0000 | 0.8818 |
| 2010014 | -0.7707 | 0.9961 | 0.0000 | 1.0000 |
| 2010023 | -0.6867 | -0.1238 | 0.0000 | 0.2323 |
| 2010032 | -0.7256 | -0.4649 | 0.0147 | 0.1650 |
| 2010038 | -0.7539 | 0.8086 | 0.0316 | 0.7475 |
| 2010040 | -0.7740 | 0.9788 | 0.0000 | 0.8806 |
| 2010049 | -0.7455 | 0.9460 | 0.0000 | 0.8324 |
| 2010057 | -0.6500 | 0.9242 | 0.0526 | 0.8011 |
| 2010079 | -0.7295 | 0.9730 | 0.0000 | 0.9382 |
| 2010096 | -0.6864 | 0.8871 | 0.0000 | 0.7487 |
| μ | -0.7362 | 0.6889 | 0.0099 | 0.7228 |
| σ | 0.0541 | 0.5272 | 0.0183 | 0.2876 |

Table B.2 shows the average ρ values, $\langle \rho_1 \rangle$ and $\langle \rho_2 \rangle$, for each experiment, per topic. It also shows the respective average Jaccard indexes, $\langle J_1 \rangle$ and $\langle J_2 \rangle$, as a complement to correlation analysis. At the end of the table, mean (μ) and standard deviation (σ) values are shown to summarize global behavior. As we can see, when comparing TF-IDF and RWS, we obtained values for $\rho_1$ that consistently approximate $-1$, with a mean of $-0.7362 \pm 0.0541$, an indication that TF-IDF and RWS are anticorrelated. If we look at $\langle J_1 \rangle$, we find extremely low similarity values between the document sets returned by TF-IDF and RWS, with this value ranging around $0.0099 \pm 0.0183$. This explains the negative correlation and, interestingly, shows that RWS can still achieve good retrieval effectiveness while returning an almost completely different set of documents than TF-IDF. Regarding the comparison of RWS with and without fatigue, we were unable to find a recurrent pattern, like we did for in the first experiment. We found both positive and negative correlations, with lower Jaccard index values associated with negative correlations. It is, however, clear that the usage of fatigue in RWS results in a different ranking than the standard RWS without fatigue. The document set similarity between the two configurations, which is of $0.7228 \pm 0.2876$, is also higher than the first experiment, which is consistent with the fact that we are testing a different version of the same ranking function. The introduction of hyperedge fatigue particularly affected topics 2010023 and 2010032 for `[ retirement age ]` and `[ japanese ballerina ]`, respectively. They both achieved a similarly low P@10 (0.0 and 0.2, versus 0.1 and 0.3, respectively, when comparing RWS with and without fatigue for the pair of topics); the behavior was similar for P@1000. Together with a low Jaccard index (0 for topic 2010023 and 0.0147 for topic 2010032) this indicates that both approaches were able to retrieve different sets of relevant documents.

## B.3 FATIGUED PAGERANK

The graph is a well-established data structure used to model a wide range of real-world relations, from social ties to protein-protein interactions, from co-authorship to the web graph, from flight patterns to time series, from term dependencies to entity relations. Network science is the area that lies between multiple domains, providing a common set of tools to study real-world networks. One of the fundamental tasks in the study of a network is the measurement of node importance or centrality, usually with the goal of obtaining an ordering or ranking. Node ranking has been a fundamental task, not only in network science, but also in information retrieval, where historical metrics like PageRank were used as query-independent evidence of the authority of a web page, in order to improve retrieval effectiveness. Many variants of PageRank have since then been developed, exploring aspects like different smoothing approaches, or applications that rely on contextual or visual features in addition to, or instead of, the traditional hyperlinks.

In this section, we propose a PageRank variant inspired by the analogy to neuronal fatigue, as briefly mentioned by von Neumann in his last lecture (Section B.1).

In particular, he said that, after being stimulated and activated, a neuron will temporarily enter a state of fatigue, during which it will not respond. Taking this characteristic as a relevant element of cognition, we decided to explore a similar idea with random walks in a graph, proposing a PageRank application where nodes have a probability of getting fatigued, in which case they are excluded from a particular step of the walk. We propose that the indegree is used as an indicator of fatigue — high indegree nodes have a higher probability of being visited and thus have a higher probability of getting fatigued. Accordingly, we combined the indegree with the transition matrix from PageRank, in order to obtain a Fatigued PageRank.

### B.3.1 Random explorer model

Random walks have been at the core of centrality metrics like PageRank, which models the behavior of a random surfer in the web graph. This means that, for each random step, there is the probability that we follow a random outgoing edge. However, given the complementary event, we simply jump to a random node in the graph. In analogy to PageRank, we propose a random explorer model to motivate and describe Fatigued PageRank.

The goal of the random explorer is to survey the graph space by randomly traversing edges, as long as it avoids recently visited nodes in order to optimize coverage — while the random surfer goes where the graph leads it, the random explorer actively tries to get to know the graph. We might say that the explorer gets fatigued and doesn't want to revisit nodes that have recently been considered — why would an explorer want to go to places it has already seen, when there is still so much to discover? Similar to PageRank, the explorer can also get stuck in sinks or cycles, or even become so fatigued that there is no interesting unexplored edge to traverse, in which case the explorer teleports to a new location to continue the survey.

In the following sections, we revise the computation of PageRank based on power iteration, introducing the notation we use and highlighting memory management via sparse matrix representations. We then extend PageRank with fatigue and propose a computation approach for Fatigued PageRank.

### B.3.2 From PageRank to Fatigued PageRank

The original PageRank corresponds to the stationary distribution of a Markov chain that models the transition probabilities between nodes in a web graph. Transitions can either happen through navigation (i.e., following a hyperlink) or through teleportation (i.e., randomly jumping to a web page). Fatigued PageRank considers similar navigation and teleportation behaviors, but introduces the concept of visitation fatigue. While in PageRank the user could only get bored and jump to another page, in Fatigued PageRank the user will also avoid recently visited pages (i.e., in the process of information seeking, the user will eventually get tired of revisiting a page and avoid it for a while). This means that memory is required to store fatigue information per node, eventually violating the Markov property (future states now depend on the current state and the fatigue state). In order to ensure that the Markov property is not violated, we approximate the probability of a node being fatigued based on its indegree — the higher the number of incoming connections, the higher the probability that a node gets fatigued, and thus the lower the probability the node is visited due to fatigue.

#### B.3.2.1 *A revision on PageRank computation using power iteration*

In order to describe Fatigued PageRank, we first illustrate the computation of PageRank based on power iteration, highlighting some simple techniques to minimize memory usage. Let us first assume a directed graph $G = (V, E)$ represented by its adjacency matrix $A$, where each row $i$ illustrates outgoing links from node

$i$ to a node $j$. Based on $A$, we need to obtain a left stochastic matrix $S$ representing the outgoing transition probabilities (each column $j$ represents the probability of transitioning from node $j$ to a node $i$). As shown in Equation B.1, this can be partly done by normalizing each column, based on the sum of its elements. However, we must use a different strategy for dealing with columns that are all zeros (representing sink nodes, without outgoing edges), since we cannot divide by zero. In Equation B.1, we simply maintain the zero-sum columns, which means that $H$ still isn't stochastic (not all columns sum to one).

$$H_{ij} = \begin{cases} 0 & \text{if } \sum_k (A^T)_{kj} = 0 \\ \frac{(A^T)_{ij}}{\sum_k (A^T)_{kj}} & \text{otherwise} \end{cases} \tag{B.1}$$

In order to obtain a stochastic matrix, despite possible sinks, we calculate matrix $S$ from matrix $H$ as described in Equation B.2.

$$S = H + \frac{1}{|V|} a e^T \tag{B.2}$$

This is done by first obtaining a binary vector $a$ that acts as a mask, where ones identify zero-sum columns. Using this mask, we then replace zero-sum columns by a uniform vector with the probability $\frac{1}{|V|}$ of randomly jumping to any node. The term $\frac{1}{|V|} a e^T$ is essentially a square matrix that repeats $\frac{1}{|V|}$ over all lines of zero-sum columns — it's as if sinks are now linked to all nodes in the graph. Finally, using Jelinek-Mercer smoothing (linear interpolation), we combine $S$ with a teleportation term, obtaining the Markov matrix $\mathcal{M}$ that is used in the power iteration to compute the PageRank vector $r$. Equation B.3 illustrates the computation of PageRank based on a damping factor $\alpha$, which is usually set to 0.85, the number of vertices $|V|$ and the column-vector $e$ of size $|V|$ and all ones. Power iteration is then initiated with any stochastic vector $r_t$, which is iteratively multiplied by $\mathcal{M}$, resulting in a normalized vector, until convergence — i.e., until $r_{t+1} \approx r_t$, as determined by the L2-norm of the difference between $r_{t+1}$ and $r_t$ and an $\epsilon$ convergence constant, frequently set to 0.001 or less.

$$\begin{aligned} r_{t+1} &= \frac{\mathcal{M} r_t}{\|\mathcal{M} r_t\|_1} = \mathcal{M} r_t \\ \mathcal{M} &= \frac{1-\alpha}{|V|} e e^T + \alpha S \end{aligned} \tag{B.3}$$

While PageRank can be computed using power iteration, as described in Equation B.3, it is easy, even for only a slightly large graph, to run out of memory during PageRank computation. This is because $\mathcal{M}$ is dense and, for dense matrices, space complexity is $O(|V|^2)$. So, for instance for a graph with 1 million nodes and assuming entries of 8 bytes, we would need 7.28 TiB of memory only to store $\mathcal{M}$, not even accounting for the overhead of the data structure. One way to mitigate this problem is to ensure that we always work with sparse matrices, that can be represented by row, column and value, only for non-zero entries. In this case, space complexity drops to $O(|E|)$, which for sparse graphs is usually a lot lower than the number of edges in a complete graph, i.e., $|E| \ll |V|^2$. This means that a graph with 10 million edges, assuming that the row, column and value each require 8 bytes to store, would only need 229 MiB of memory to be stored. Since matrix $H$ is sparse, we can simply allocate space for $H$ using a sparse matrix representation, for the zero-sum mask vector $a$, for the vector $e$ of ones, and for the PageRank vector $r$, and we

can then compute matrix $\mathcal{M}$ on-the-fly during power iteration cycles as shown in Equation B.4.

$$r_{t+1} = \left(\alpha H + (\alpha a + (1-\alpha)e)\left(\frac{1}{|V|}e^{\top}\right)\right) r_t \tag{B.4}$$

#### B.3.2.2 *Fatigued PageRank computation using power iteration*

Like PageRank, Fatigued PageRank also considers the idea of teleportation as a way to avoid sinks or cycles, leaving the corresponding term of the equation unchanged. Unlike PageRank, the navigation term is not only based on the outgoing transitions $H$, but also on a fatigue-derived factor. In particular, we approximate the probability of fatigue of a node based on its indegree vector $k^{-}$. We then normalize $k^{-}$ based on the maximum possible indegree $|V|-1$ (ignoring loops), while using additive smoothing in order to avoid completely removing transitions to nodes with a maximum probability of fatigue. We use a low impact smoothing constant $\beta = 0.1$, mainly just to avoid zeros — a zero would completely block a transition, while a small probability will provide a chance for the transition to have an effect during power iteration. Finally, we combine $H$ with the complement of the normalized and smoothed indegree, after renormalizing the vector to ensure the it remains stochastic. We do this through element-wise multiplication ($\odot$) with each column $j$ of $H$, as show in Equation B.5.

$$\begin{aligned} k^{*} &= 1 - \frac{k^{-} + \beta}{|V| - 1 + \beta} \\ H'_{\cdot j} &= \frac{k^{*}}{\|k^{*}\|_1} \odot H_{\cdot j} \end{aligned} \tag{B.5}$$

Essentially, a node becomes less probable to visit, if it has a high probability of getting fatigued. Fatigued PageRank is then computed based on power iteration, as defined in Equation B.3, after replacing $H$ with $H'$ in the computation of $S$ (Equations B.2 and B.4).

#### B.3.2.3 *Exemplifying Fatigued PageRank computation*

In this section, we illustrate the calculation of Fatigued PageRank, using the toy example graph in Figure B.2. We prepared a graph with two sources (nodes 1 and 4 only have outgoing links) and a sink (node 5 only has incoming links). Sink nodes serve to illustrate the need for a teleportation term in PageRank (or Fatigued PageRank), while source nodes serve to illustrate the effect of minimum fatigue — as we can see in $k^{*}$, both nodes 1 and 4 have maximum probability ($k^{*}_1 = k^{*}_4 = 0.26$). We also attempted to include a node with the maximum number of inlinks (i.e., $|V|-1$ for a graph without loops). However, in order to keep the toy example clean and to ensure we had at least one sink, we opted to only include node 3 with a high indegree of $|V|-2$. Notice, however, that a node with indegree $|V|-1$ would result in a normalized indegree of one and therefore correspond to a zero entry in $k^{*}$ for $\beta = 0$ — that is, without additive smoothing, transitions to nodes with maximum indegree would be completely blocked (we use $\beta = 0.1$).

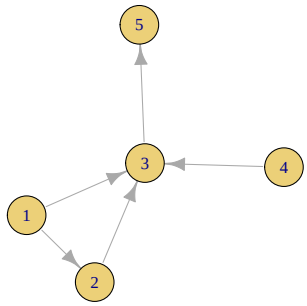



**Figure B.2:** Toy graph, with one source (node 4) and one sink (node 5)

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

$$H = \begin{bmatrix} 0.00 & 0.00 & 0.00 & 0.00 & \mathbf{0.00} \\ 0.50 & 0.00 & 0.00 & 0.00 & \mathbf{0.00} \\ 0.50 & 1.00 & 0.00 & 1.00 & \mathbf{0.00} \\ 0.00 & 0.00 & 0.00 & 0.00 & \mathbf{0.00} \\ 0.00 & 0.00 & 1.00 & 0.00 & \mathbf{0.00} \end{bmatrix}$$

$$a^T = \begin{bmatrix} 0 & 0 & 0 & 0 & \mathbf{1} \end{bmatrix}$$

As we can see, we begin with the adjacency matrix $A$, which we transform into $H$ by transposing and normalizing columns that are not all zeros. Zero-sum columns are then identified by a 1 in the corresponding position of vector $a$. The corresponding row of $A$, column of $H$ and value of $a$ are all displayed in bold for a clearer understanding of such a process. While the zero-sum columns of $H$ could be have been replaced by the teleportation probability, we avoid doing so to save memory, taking better advantage of a sparse matrix representation. We instead do all computations on-the-fly during power iteration, resulting in a low memory footprint, since $H$ is static and only the PageRank vector must be updated (which can even be done in-place to further save memory). We can then set the damping factor to $\alpha = 0.85$, allocate a vector $e$ of ones, calculate the probability of teleportation $\frac{1}{|V|} = 0.20$ and calculate PageRank using Equation B.4 by initializing $r_0$ for instance to the uniform probability 0.20. For large graphs, the computation can even be done using blocks of rows of $H$, along with the corresponding elements of $a$ and $e$ ($e^T$ remains untouched, though). This results in incremental blocks of $r_t$ that can be sequentially concatenated and even processed in parallel.

$$[k^*]^T = \begin{bmatrix} 0.26 & 0.20 & 0.08 & 0.26 & 0.20 \end{bmatrix}$$

$$H' = \begin{bmatrix} 0.00 & 0.00 & 0.00 & 0.00 & \mathbf{0.00} \\ 0.71 & 0.00 & 0.00 & 0.00 & \mathbf{0.00} \\ 0.29 & 1.00 & 0.00 & 1.00 & \mathbf{0.00} \\ 0.00 & 0.00 & 0.00 & 0.00 & \mathbf{0.00} \\ 0.00 & 0.00 & 1.00 & 0.00 & \mathbf{0.00} \end{bmatrix}$$

$$r_0^T = \begin{bmatrix} 0.20 & 0.20 & 0.20 & 0.20 & 0.20 \end{bmatrix}$$
$$r_1^T = \begin{bmatrix} 0.03 & 0.15 & 0.42 & 0.03 & 0.37 \end{bmatrix}$$
$$\dots$$
$$r_{10}^T = \begin{bmatrix} 0.05 & 0.09 & 0.23 & 0.05 & 0.59 \end{bmatrix}$$

For Fatigued PageRank, however, we also need to compute $k^*$, which will by multiplied by each column of $H$ to generate $H'$ as shown above. Power iteration will then incrementally update $r_t$ until convergence — we show the values for $r_0$, $r_1$ and $r_{10}$ to illustrate how conversion happens in only 10 iterations.

### B.3.3 Performance as a node ranking metric

In order to assess the quality of Fatigued PageRank as a node ranking metric, we used an evaluation strategy similar to Dimitrov et al. [157], based on a Wikipedia's link graph, annotated with the number of transitions from its clickstream as the ground truth. This was the main reason for creating the Simple English Wikipedia Link Graph with Clickstream Transitions 2018-12, which we described in Section 4.2.1.

We computed the indegree, HITS authority, PageRank and Fatigued PageRank for the link graph. The top 5 pages according each metric are shown in Table B.3. As we can see, all rankings are distinct, with *United States* being the only common entity across indegree, PageRank and Fatigued PageRank. On the other hand, HITS

**Table B.3:** Top 5 nodes according to each node ranking metric for Simple English Wikipedia.

**(a)** Ranking by indegree and HITS authority.

| | **Indegree** | **HITS Authority** |
|---|---|---|
| 1 | *United States* | *Lisa Bonet* |
| 2 | *France* | *Road to Paloma* |
| 3 | *International Standard Book Number* | *Ronon Dex* |
| 4 | *Geographic coordinate system* | *Native Hawaiians* |
| 5 | *Americans* | *Aquaman* |

**(b)** Ranking by PageRank and Fatigued PageRank.

| | **PageRank** | **Fatigued PageRank** |
|---|---|---|
| 1 | *United States* | *United States* |
| 2 | *United Kingdom* | *International Standard Book Number* |
| 3 | *India* | *France* |
| 4 | *List of United States cities by population* | *United Kingdom* |
| 5 | *Periodic table* | *City* |

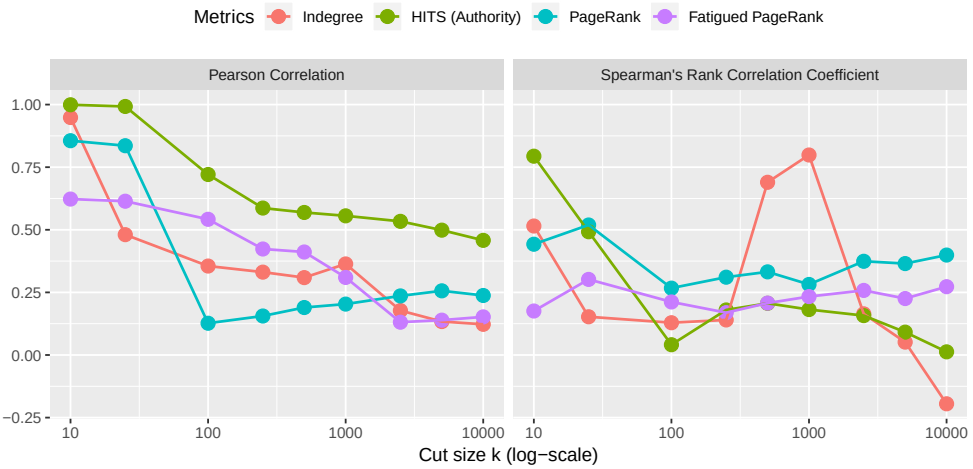



**Figure B.3:** Comparison of node ranking metrics with the number of visits, given by the sum of incoming transitions from the Wikipedia clickstream dataset.

authority contains a notably different list of entities when compared to the remaining rankings, which is more related to entertainment, namely representing actors, movies and Hawaii (possible in reference to *Jason Momoa*, who plays the *Ronon Dex* character).

For each metric, we then computed its Pearson and Spearman correlations with the number of visits (i.e., the sum of incoming transitions). This follows the strategy used by Dimitrov et al. [157] in their study of the Weighted PageRank, however we used a much smaller Wikipedia graph. We are particularly interested in the Spearman correlation, which compares rankings, but we also included the Pearson correlation as a way to characterize behavior. We analyzed the 897,577 Simple English Wikipedia articles. The best overall results were obtained for PageRank, which achieved a Spearman correlation of 0.9902, followed by HITS authority with 0.6330, indegree with 0.3920 and, only then, Fatigued PageRank with 0.1604.

The low results for Fatigued PageRank led us to investigate further, by looking at the correlation for different ranking cuts of size $k$, according to each metric, as illustrated in Figure B.3. In particular, we plotted the correlation coefficients for $k \in \{10, 25, 100, 250, 500, 1000, 2500, 5000, 10000\}$ using a log-scale. This means that, for instance for PageRank, we sorted from highest to lowest PageRank and

**Table B.4**: Variance of the correlations over all cuts, in ascending order by the variance of Spearman's rank correlation.

| Metric | Variance | |
|---|---|---|
| | Pearson | Spearman |
| Fatigued PageRank | 0.0399 | 0.0019 |
| PageRank | 0.0826 | 0.0064 |
| HITS (Authority) | 0.0421 | 0.0624 |
| Indegree | 0.0631 | 0.1052 |

kept the top k values, which we then correlated with the corresponding number of visits. When looking at Spearman correlation, we found that, for the top 10, we obtained the best results using HITS authority. We also found that, for the top 10, the indegree is a good approximation of PageRank, which is consistent with the literature [407]. As the cut size increases, though, the quality of HITS authority decreases substantially (it had the lowest coefficient for the top 100), remaining low but fairly stable until the top 10000. Another interesting characteristic we found was the high variability of the quality of the indegree as a node ranking metric.

As we can see in the figure, the Spearman correlation for the indegree suddenly increases at around top 1000, only to sharply drop after that. For that cut, we found that there were several ties where the indegree for most nodes was around 705, while the corresponding number of visits was also tied with values around zero — this was responsible for the sudden boost in correlation. However, when further ahead the wrongly ranked nodes are added to the top, the correlation sharply drops to a negative coefficient. This variability shows the "naïveness" of the indegree.

Out of all the metrics, only PageRank and Fatigued PageRank have shown to be consistently stable for different cut sizes, however Fatigued PageRank appears to be consistently lower than PageRank. This can be explained by the fact that, inherently, fatigue will take away from visiting the most popular nodes. We hypothesize that fatigue introduces diversity and somewhat "redistributes the wealth", potentially providing a better way to explore the graph, by reducing the bias of popularity. This, however, requires further investigation and is out of the scope of this paper.

As shown in Table B.4, we inspected the variance of the overall correlation values, in order to understand the consistency of each metric for increasing cut sizes. We found that Fatigued PageRank obtained the lowest variances for the Pearson and Spearman's rank correlations, showing robustness to varying the cut size. When considering Pearson's correlation, PageRank obtained the highest (worst) variance, while, for Spearman's rank correlation, it obtained the second lowest (best) variance. On the other side, as expected, the indegree had the highest (worst) variance in both cases, illustrating its lack of robustness.

### B.3.4 Performance as a query-independent feature in search

In this section, we evaluate the impact of Fatigued PageRank in retrieval effectiveness, when combined with BM25, in comparison with other graph-based metrics. These metrics were computed over the companion link graph built from TREC Washington Post Corpus, using the inter-document hyperlinks found in HTML anchors, as described in Section 4.1.2.1.

We computed the indegree, HITS authority, PageRank and Fatigued PageRank for the TREC Washington Post Corpus link graph. The top 5 documents (news articles or blog posts) according to each metric are shown in Table B.5. The behavior is similar to the rankings for the Simple English Wikipedia (cf. Section B.3.3), where each metric results in a distinct ranking. The indegree and HITS authority share two common titles, while the indegree and PageRank, as well as PageRank and Fatigued PageRank, share one common title. Once again, HITS authority contains a large

**Table B.5:** Top 5 nodes according to each node ranking metric for TREC Washington Post Corpus.

**(a)** Ranking by indegree and HITS authority.

| | **Indegree** | **HITS Authority** |
|---|---|---|
| 1 | *Trump recorded having extremely lewd conversation about women in 2005* | *This photo of an officer comforting a baby went viral. But there's more to the story.* |
| 2 | *This professor has predicted every presidential election since 1984. He's still trying to figure out 2016.* | *'Heartbreaking' video captures toddler trying to wake mother after apparent overdose* |
| 3 | *This photo of an officer comforting a baby went viral. But there's more to the story.* | *Ohio city shares shocking photos of adults who overdosed with a small child in the car* |
| 4 | *Trump is headed for a win, says professor who has predicted 30 years of presidential outcomes correctly* | *The heroin epidemic's toll: One county, 70 minutes, eight overdoses* |
| 5 | *'Heartbreaking' video captures toddler trying to wake mother after apparent overdose* | *At 18 years old, he donated a kidney. Now, he regrets it.* |

**(b)** Ranking by PageRank and Fatigued PageRank.

| | **PageRank** | **Fatigued PageRank** |
|---|---|---|
| 1 | *Trump recorded having extremely lewd conversation about women in 2005* | *Here's a guide to the sex allegations that Donald Trump may raise in the presidential debate* |
| 2 | *78 Republican politicians, donors and officials who are supporting Hillary Clinton* | *The facts about Hillary Clinton and the Kathy Shelton rape case* |
| 3 | *An unlikely Bush finally did some damage to Donald Trump: Billy Bush* | *Khizr Khan's loss: A grieving father of a soldier struggles to understand* |
| 4 | *Professor who predicted 30 years of presidential elections correctly called a Trump win in September* | *The father of Muslim soldier killed in action just delivered a brutal repudiation of Donald Trump* |
| 5 | *Here's a guide to the sex allegations that Donald Trump may raise in the presidential debate* | *Obamas sign book deals with Penguin Random House* |

**Table B.6:** Retrieval effectiveness of graph-based metrics, as query-independent evidence, when combined with BM25 using the *sigm* function.

| **Model** | **GMAP** | **MAP** | **NDCG@10** | **P@10** |
|---|---|---|---|---|
| BM25 | 0.1395 | 0.2031 | 0.3528 | 0.3700 |
| BM25 + Indegree | 0.1357 | 0.1994 | 0.3537 | 0.3800 |
| BM25 + HITS Authority | 0.1395 | 0.2031 | 0.3540 | 0.3720 |
| BM25 + PageRank | 0.1395 | 0.2031 | 0.3528 | 0.3700 |
| BM25 + Fatigued PageRank | 0.1395 | 0.2031 | 0.3528 | 0.3700 |

fraction of potentially-viral titles, showing its ability to rank from an entertainment point of view — in Wikipedia we had found movies and actors in the top ranks according to HITS authority.

In search, we can use graph-based features as query-independent evidence (e.g., PageRank) that can be combined with other retrieval models (e.g., BM25). This can be done for instance through a linear combination, as a feature in a learning-to-rank model, or through a post-processing reranking approach. We adopt a more classical approach for combining query-dependent and query-independent features, as described by Craswell et al. [170] and conveniently available in Apache Lucene[1] since version 7.4.0. Craswell et al. explored reranking approaches, experimenting

1 https://lucene.apache.org/

**Table B.7:** Top 5 most frequent document titles in TREC Washington Post Corpus.

| Freq. | Title |
|---|---|
| 455 | *Happy Hour Roundup* |
| 353 | *A 7-year-old told her bus driver she couldn't wake her parents. Police found them dead at home.* |
| 320 | *How long before the white working class realizes Trump was just scamming them?* |
| 310 | *Five dead teens, a stolen cop car and the 'most horrific' crash in decades* |
| 252 | *Trump is headed for a win, says professor who has predicted 30 years of presidential outcomes correctly* |

with BM25 as the baseline and PageRank as the query-independent graph-based evidence, while studying statistical dependence. PageRank was converted into a relevance weight and added to BM25 after applying one of three proposed functions: *log*, *satu* or *sigm*. Blanco and Lioma [15, §5.2.5] have also used this type of integration for query-independent graph-based features, specifically exploring the *satu* function. In this experiment, we use the *sigm* function, with the parameters that generated the best MAP in the Craswell et al. experiments ($w = 1.8$, $k = 1$, $a = 0.6$). This enabled us to evaluate Fatigued PageRank's impact in retrieval effectiveness, when compared to alternative metrics.

Table B.6 shows the results for this evaluation, where we used BM25 as the baseline and computed classic metrics, like MAP, NDCG@10 and P@10, for measuring effectiveness. We also included GMAP, which is less sensitive to outliers than MAP, since it uses the geometric mean instead of the arithmetic mean to aggregate the average precisions — when GMAP is lower than MAP, it usually means that only a few topics were driving up the score, despite most of them actually having a lower average precision than MAP would lead us to believe. The results show that the overall impact of the graph-based metrics is minimal, except for the indegree, which decreases effectiveness for GMAP and MAP, but increases effectiveness for NDCG@10 and P@10, which only consider the top 10 results. This is consistent with the results shown in Figure B.3, where the indegree is shown to be slightly better than PageRank, but only for the top 10.

The results were quite neutral regarding the overall benefits of graph-based metrics to improve retrieval effectiveness over the TREC Washington Post Corpus. We hypothesize that, despite their unique identifiers and URLs, the high number of pages with a duplicate title (cf. Table B.7) affected both the text-based relevance score and the graph-based relevance score. It introduced noise that, by being removed, could have provided a better insight into the effect of the studied query-independent evidence. This would, however, require a different set of relevance judgments that are not currently available.

## SUMMARY

Inspired by von Neumann's last lecture and his motivation for crossover work between computer science and neuroscience, we have proposed the application of fatigue to random walks in hypergraphs and for the computation of PageRank. After stimulated, a neuron in the brain enters a state of fatigue that lasts a given period of time. We applied this analogy to nodes and hyperedges where, during a random walk, a node and/or hyperedge would be fatigued for a given number of cycles before a random walker could traverse it again.

For the random walk score, we found that fatigue was able to significantly improve retrieval efficiency, at the cost of effectiveness, particularly when compared to the RWS version without fatigue. While we were unable to surpass the baselines, we were able to introduce fatigue in hyperedges and achieve a similar performance to the random walk score without fatigue. Through correlation analysis, we further investigated the similarities between the rankings obtained from different models. We specifically compared the TF-IDF baseline with the best RWS with fatigue. We found that RWS was able to effectively retrieve documents, reaching a comparable performance to the baselines, but returning a different set of relevant documents, in a nearly anticorrelated manner. Using a similar strategy, we also compared two configurations of RWS, with and without fatigue, also finding that they returned different sets of relevant documents, while showing an overall positive correlation, except for two topics, where only a few relevant results could be retrieved.

We also applied the idea of fatigue to PageRank, formalizing a novel Fatigued PageRank metric that follows a random explorer model, being analogous to the combination of PageRank and Reverse PageRank, or the authority and hub scores from HITS. We then evaluated our graph-based metric, comparing it with the indegree, HITS authority and PageRank, when computed over the Simple English Wikipedia link graph. Based on Spearman's rank correlation coefficient, we assessed the quality of each metric in comparison with the number of user visits. We found that, while Fatigued PageRank obtained the lowest overall correlation of 0.1604, it was actually able to outperform the indegree and HITS authority for the top 10,000 nodes, showing a more consistent behavior than both those metrics, with the lowest variance for the Spearman's rank correlations of all metrics. Despite having a lower overall correlation score than PageRank, the fatigued version might work in favor of bias reduction towards the most popular nodes, ensuring diversity in the preparation of the ranking.

We also assessed the impact in retrieval effectiveness of Fatigued PageRank when used as query-independent evidence, in combination with a text-based relevance score like BM25. We used the sigmoid function proposed by Craswell et al. to transform the graph-based feature into a static relevance weight that could be used as an additional signal during ranking. Based on the TREC Washington Post Corpus, we were unable to find significant differences in the retrieval effectiveness, except for the indegree metric, which decreased GMAP and MAP, but increased NDCG@10 and P@10, over a BM25 baseline. We believe that the existence of duplicate documents in the collection was detrimental to this experiment, given the positive results with graph-based metrics found in the literature (e.g., Najork [408, Figure 3]). Another possible explanation would be regarding the specificity of the queries used for evaluation, given that general queries usually benefit from graph-based metrics, while specific queries do not (cf. Najork [408, Figure 4]).

# C OVERVIEW OF ENTITY-ORIENTED SEARCH APPROACHES AND TASKS

## C.1 CLASSICAL MODELS

**Table C.1:** Classical information retrieval models applied to entity-oriented search.

| Approach | Task(s) | Description |
|---|---|---|
| Virtual documents | Ad hoc entity retrieval | **Bautin and Skiena [4].** Time-dependent concordances for entities (i.e., concatenation of sentences containing the entity, optionally for a given period of time).<br>**Dietz et al. [131].** Knowledge portfolios used to establish query-specific collections of entities and text passages relevant to the queries. Retrieval based on textual passages or Wikipedia pages for the entities in the query. |
| | Ad hoc entity retrieval; Attribute retrieval; Relation retrieval | **Pound at al. [3].** Defined five query categories: entity, type, attribute, relation and keyword. Index RDF using an inverted index, computing IDF per RDF property, as opposed to the whole collection. Used TF-IDF for ranking. |
| Combined data | Ad hoc document retrieval; Ad hoc entity retrieval | **Bhagdev et al. [123].** Documents identified by a URI and indexed using an inverted index. Entities stored in a triple-store with provenance linking to document URIs. Their hybrid search approach consisted of either document retrieval informed by entities, entity retrieval informed by documents, or a combination of both.<br>**Bast and Buchhold [74].** Joint index for ontologies and text, based on context lists and ontology relation lists. Context lists map words or entities to text postings, by their prefixes, while ontology relation lists map source entities to target entities, along with an optional relation score. |
| | Ad hoc document retrieval; Ad hoc entity retrieval; Related entity finding | **Zhou [130].** Querying by entities: entities as input and documents or entities as output; entities represented by their Wikipedia pages. Querying for entities: keywords or entities as input and entities as output; proposed the CQL language over a joint index and a contextual index. Querying by entities and for entities: entities as input and output; proposed a framework analogous to related entity finding [62, §4.4.3]. |
| | Entity list completion | **Bron et al. [128].** In order to retrieve related entities, they proposed three approaches: text-based (using the given textual description as input), structure-based (using the the given example entities as input) and a combination of both, which outperformed one isolated. |
| | – | **See also:** Tonon et al. [409], Xion et al. [257]. |
| Probabilistic graphical models | Ad hoc entity retrieval | **Koumenides and Shadbolt [125].** Bayesian network to establish dependencies between entities and property instances, between property instances and property identifiers and, finally, between terms in the literal space and property identifiers. Entity search carried through Bayesian inference<br>**Raviv et al. [124].** Markov network to model the undirected dependencies between the query and the entity. Captured the dependencies between a virtual document (representing the entity) and the query, between the entity type and the query target type, and between the entity name and the query. |
| | Sentence retrieval | **Urbain [126].** Markov network to model the undirected dependencies between the query and the sentence. Several models were tested, with different feature functions: aggregate (entity, sentence terms, document terms), term (sentence term, document term), entity, entity-relation and relation. |
| Cluster hypothesis | Ad hoc entity retrieval | **Raviv et al. [73].** Verified the cluster hypothesis for entity-oriented search: closely related entities have a high probability of also being relevant to the query. This is important for instance when implementing graph-based approaches that rely on distance. |

## C.2 LEARNING-TO-RANK MODELS

**Table C.2:** Learning to rank models for entity-oriented search.

| Approach | Task(s) | Description |
|---|---|---|
| Semantic-driven | Sentence retrieval | **Chen et al. [137].** Explored explicit semantic analysis (ESA) and word2vec skip-gram as query-sentence similarity features. Used Metzler-Kanungo features as the baseline and experimented with linear regression, coordinate ascent and MART. Combining all features yielded the best results, with ESA distinguishing itself positively. |
| | Related entity finding | **Lin et al. [141].** Retrieved the source entity homepages based on a given narrative illustrating the relation to a target entity of a given type. Applied entity extraction, obtaining candidate target entities and computed several source-target entity-entity features, like frequency, proximity and semantic similarity. Experimented with three SVMs: (i) using default hyperparameters, (ii) using tuned hyperparameters, and (iii) using feature selection. |
| | Ad hoc entity retrieval | **Schuhmacher et al. [143].** Given a keyword query, ad hoc entity retrieval was implemented through: (i) document ranking; (ii) entity linking; and (iii) entity ranking. Features included mention frequency, as well as query-mention, query-entity and entity-entity similarities. A semantic kernel was used for the latter. Learning to rank models slightly improved individual feature baselines. |
| Virtual documents | Ad hoc entity retrieval | **Chen et al. [144].** Compared a fielded sequential dependence model (FSDM; baseline) with pairwise (RankSVM) and listwise (coordinate ascent) methods. Features included a language model, BM25, coordinate match, cosine similarity, SDM and FSDM. Results were consistently better for learning to rank over several test collections. They also found related entity names to be a fundamental field, except for question answering, highlighting the importance of training several models per query type. |
| Representation learning | Ad hoc entity retrieval | **Gysel et al. [86].** Tackled the keyword-based entity retrieval problem by learning a common embedding for words and entities, called latent semantic entity (LSE). They then used learning to rank based on the embeddings, but the embeddings could just as easily be applied to the vector space model. Their best feature configuration included LSE, along two other features. |
| – | – | **See also:** Reinanda et al. [110] |

## C.3 GRAPH-BASED MODELS

**Table C.3**: Graph-based models for entity-oriented search.

| Approach | Task(s) | Description |
|---|---|---|
| Link analysis* | Node importance | **Kleinberg [149].** Hypertext Induced Topic Selection (HITS) provides two scores for a node: hub and authority. The hub score is higher when a node links to multiple nodes or to highly authoritative nodes. The authority score is higher when a node receives multiple links or those links are from well-renowned hubs. HITS is usually computed for a query-dependent graph.<br>**Page and Brin [113, 151].** PageRank measures the importance of a node based on the importance (and number) of incoming nodes. PageRank is usually computed for a query-independent graph (e.g., web graph). |
| | Node relatedness | **Van and Beigbeder [162].** Explored bibliographic coupling (shared outgoing links) and co-citation (shared incoming links) as reranking strategies. In practice, two nodes were related based on the similarity of their immediate neighborhood. |
| | Node importance;<br>Node relatedness | **Ito et al. [163].** Explored von Neumann kernels as a unified framework for measuring importance and relatedness. Also proposed Laplacian kernels and heat kernels as a way to control the bias between relatedness and importance, and to tackled limitations of bibliographic coupling and co-citation approaches. |
| | Node importance;<br>Graph partitioning | **Chung [164].** Proposed the heat kernel PageRank, building on PageRank's alternative notation [166, §1.5]. Local applications can be used to identify node clusters, while global applications to measure node importance.<br>**Kloster and Gleich [165].** Explored the heat kernel PageRank as a community detection algorithm, solving the exponential of the Markov matrix using a Taylor polynomial approximation.<br>**Yang et al. [169].** Proposed a more efficient approach to computing heat kernel PageRank, based on Monte Carlo random walks and a reduction of the required number of random walks. |
| | – | **See also:** Kandola et al. [410]. |
| Text as a graph** | Ad hoc document retrieval | **Blanco and Lioma [15].** Document as an undirected graph of co-occurring words within a sliding window, or with an added direction based on Jespersen's rank theory of POS tags. They experimented with PageRank and indegree over the two graphs as a TF replacement, combining the score with global graph-based features (e.g., average degree). Performance was improved over TF-IDF and BM25.<br>**Rousseau and Vazirgiannis [16].** Similar to Blanco and Lioma [15], they defined a graph-of-word of co-occurring words, but considered direction and the following terms instead of centering the sliding window on each word. They found little impact of window size (used $N = 4$) and used almost no pivoted document length normalization ($b = 0.003$). |
| | Ad hoc document retrieval;<br>Text classification | **Dourado et al. [171].** Represented documents as a bag of textual graphs, weighting unigrams and bigrams by term frequency. A graph dissimilarity function was proposed to cluster subgraphs and obtain a graph-based vocabulary. Assignment to this vocabulary resulted in a matrix (each subgraph compared to all centroids), which was then collapsed into a vector to represent the document, as a pooling of graph embeddings. The resulting embedding could then be used for text retrieval and classification. |
| Knowledge graphs | Ad hoc entity retrieval;<br>Ad hoc document retrieval | **Fernández et al. [172].** Given a natural language query, triples were retrieved and, in turn, used to retrieve and rank documents, based on a semantic index that combined information from a knowledge graph and a corpus.<br>**Balog et al. [175].** Used keyword queries and language models to retrieve documents and entities from news collections. They defined both a document-centric and an entity-centric view on their SaHaRa system, where entities augmented documents and vice-versa. |

(Continued on next page)

Table C.3. Graph-based models models for entity-oriented search. (Continued from previous page)

| Approach | Task(s) | Description |
|---|---|---|
| Knowledge graphs (cont.) | Ad hoc entity retrieval | **Blanco et al. [176].** Experimented with several ways to translate an RDF graph into a multi-fielded index: horizontal (*token*, *property*, *subject*), vertical (one field per property), and reduced-vertical (*important*, *neutral* and *unimportant* groupings of properties). Ranking was based on BM25F.<br>**Neumayer et al. [179].** Experimented with two entity models to translate an RDF graph into a multi-fielded index: unstructured (one field for all properties) and structured (four property groups: *Name*, *Attributes*, *OutRelations*, *InRelations*). Ranking was based on language models. |
| | Knowledge graph construction and modeling | **Byrne [174].** Used RDF to integrate structured data from relational databases (each table was considered a class), with domain thesauri (represented using the SKOS ontology), and free text (using NER to identify 11 classes of entities, and relation extraction to identify 7 predicates). She compared retrieval based on SPARQL and SQL.<br>**Google Knowledge Graph [5].** Announced in 2012 and partly powered by Freebase. Freebase was then bought and closed by Google. Public dumps were made available and migrated to Wikidata.<br>**Microsoft Satori [54].** Announced in 2013 and presented in KDD 2018 as a tutorial on building knowledge graphs. Focused on evaluation by correctness, coverage, freshness and usage.<br>**Microsoft Academic Graph, by Sinha et al. [186].** It contains 80 million indexed papers and six types of entities: *#field_of_study*, *#author*, *#institution*, *#paper*, *#venue* and *#event*. Built to support academic queries, based on feeds from publishers and event web sites. |
| | Topic modeling | **Allahyari [254].** Proposed a method for ontology-based topic modeling, experimenting with topics as distributions over ontology concepts, as well as topics as distributions over Wikipedia categories. |
| | – | **See also:** Gao et al. [185] |
| Text to entity graph | Ad hoc entity retrieval | **Bordino et al. [187]** Serendipitous search over an entity graph extracted from Wikipedia and Yahoo! Answers. |
| | Ad hoc document retrieval | **Ni et al. [189].** Measuring semantic similarity based on a concept graph representing a document. Proposed *Concept2VecSim* and *ConceptGraphSim*. |
| Entity graph to tensor | Ad hoc entity search | **Zhiltsov and Agichtein [75].** Represented entities as a tensor of adjacency matrices (one per predicate). Using tensor factorization, they obtained a matrix of latent entity embeddings, that they used to compute similarities to the top-3 entities, boosting those entities (consistent with the cluster hypothesis). |
| Graph matching | Ad hoc entity retrieval | **Zhu et al. [58] and Zhong et al. [57].** Matched a query graph with an entity graph (conceptual graph; also translatable to RDF). They computed the semantic similarity based on the similarity between the nodes and edges of two conceptual graphs. The user was required to provide a set of entry nodes as part of the query.<br>**Zhu et al. [199].** Translated a natural language query into a graph query using named entity recognition along with dependency parsing to extract entities and their relations. The result was translated into a graph query language for a graph database. |
| | Ad hoc entity retrieval; Related entity finding; Entity list completion | **Minkov and Cohen [196].** Generalized multiple personal information management tasks over an entity graph and based on keyword queries (e.g., name disambiguation, threading, grouping e-mail aliases). |
| | Answer tree ranking | **Zhong et al. [198].** Combined content-based and structure-based features to score answer trees that contained query keywords. |
| Hypergraph-based* | Joint representation | **Garshol [203].** Describes topic maps, a hypergraph of topics, their associations and occurrences. It describes it as a common reference model, able to represent controlled vocabularies, taxonomies, thesauri, faceted classification and ontologies.<br>**Yi [204].** Compared thesaurus based information retrieval with topic maps based information retrieval, finding topic maps to outperform thesauri. |

(Continued on next page)

Table C.3. Graph-based models models for entity-oriented search. (Continued from previous page)

| Approach | Task(s) | Description |
|---|---|---|
| Hypergraph-based* (cont.) | Ad hoc document retrieval | **Bendersky and Croft [14].** Proposed the query hypergraph to represent higher-order dependencies between concepts (subsets of query terms) and a document. Ranking is done using a log-linear combination of factors, based on the factor graph representation of the hypergraph. |
| | Ad hoc entity retrieval; Joint representation | **Dietz [103].** Proposed ENT Rank for modeling entity-neighbor-text relations as a hypergraph. She transformed the hypergraph into an entity co-occurrence multigraph which was used to determine which features, from text, entites, and their relations, to combine for learning a function for entity ranking. |
| | Document representation | **Haentjens Dekker and Birnbaum [76].** Text As a Graph (TAG) is a document representation based on a hypergraph. It links text, document, annotation and markup nodes. It can for instance be used to represent a poem line or quatrain as hyperedges of text, where the quatrains subsume lines. |
| | – | **See also:** Xiong and Ji [206], Cattuto et al. [207], Bu et al. [83], McFee and Lanckriet [210], Tan et al. [209], Theodoridis et al. [211], Bellaachia and Al-Dhelaan [55], Lee-Kwang and Lee [213], Akram and Dudek [214] |
| Random walk based | Ad hoc entity retrieval | **Hogan et al. [66].** ReConRank is used to rank nodes in a query-dependent graph, that jointly represents RDF resources and contexts.<br>**Balmin et al. [65].** ObjectRank is used to rank nodes in a query-dependent labeled graph. A graph is induced by each query term and PageRank is used to compute a term-based score (i.e., personalized by a term) along with a global score (i.e., without personalization). An authority transfer schema is used to introduce edge bias.<br>**Chakrabarti [67].** HubRank provides a more efficient alternative to ObjectRank. It is based on precomputed random walk fingerprints over a subgraph limited by a set of boundary nodes (blockers and losers). |
| | Node importance | **Espín-Noboa et al. [239].** HopRank models human navigation on semantic networks, by taking into consideration the bias of jumping to nodes at particular distances. Node importance is adjusted accordingly.<br>**Nie et al. [235].** PopRank assigns node importance based on information from an entity graph (object graph) and a context graph (web graph). It was used in Libra, which is now Microsoft Academic Search.<br>**Delbru et al. [72].** DING (Dataset rankING) also assigns node importance based on information from an entity graph and a context graph (dataset graph based on entity links). |
| | Related entity finding; Entity list completion | **Musto et al. [238].** Proposed a semantics-aware personalized PageRank for recommendation over a user-item-entity graph, built by combining a user-item profile with DBpedia triples. This is similar to the tasks of related entity finding or entity list completion, if the user is abstracted as an entity. |

* General (hyper)graph-based approaches and introductory concepts that can be applied to entity-oriented search (e.g., multiple PageRank adaptations to entity-oriented search are covered in Section 2.2.8, and the query hypergraph, which defines concepts that can be entities, is covered in Section 2.2.7).

** Despite not leveraging entities, these models are relevant when defining joint graph-based representations for text and entities.

# D | PUBLISHED WORK

**2020**

J. Devezas and S. Nunes. "Characterizing the hypergraph-of-entity and the structural impact of its extensions". In: *Applied Network Science - Special Issue of the 8th International Conference on Complex Networks and Their Applications* 5.1 (2020), p. 79. ISSN: 2364-8228. DOI: 10.1007/s41109-020-00320-z

J. Devezas. "Graph-Based Entity-Oriented Search: A Unified Framework in Information Retrieval". In: *Advances in Information Retrieval - 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14-17, 2020, Proceedings, Part II*. ed. by J. M. Jose, E. Yilmaz, J. Magalhães, P. Castells, N. Ferro, M. J. Silva, and F. Martins. Vol. 12036. Lecture Notes in Computer Science. Springer, 2020, pp. 602–607. DOI: 10.1007/978-3-030-45442-5_78

J. L. Devezas and S. Nunes. "Army ANT: A Workbench for Innovation in Entity-Oriented Search". In: *Advances in Information Retrieval - 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14-17, 2020, Proceedings, Part II*. ed. by J. M. Jose, E. Yilmaz, J. Magalhães, P. Castells, N. Ferro, M. J. Silva, and F. Martins. Vol. 12036. Lecture Notes in Computer Science. Springer, 2020, pp. 449–453. DOI: 10.1007/978-3-030-45442-5_56

**2019**

J. Devezas and S. Nunes. "Hypergraph-of-entity: A unified representation model for the retrieval of text and knowledge". In: *Open Computer Science* 9.1 (June 2019), pp. 103–127. DOI: 10.1515/comp-2019-0006

J. Devezas and S. Nunes. "Characterizing the Hypergraph-of-Entity Representation Model". In: *Complex Networks and Their Applications VIII - Volume 2 Proceedings of the Eighth International Conference on Complex Networks and Their Applications COMPLEX NETWORKS 2019, Lisbon, Portugal, December 10-12, 2019*. 2019, pp. 3–14. DOI: 10.1007/978-3-030-36683-4_1

J. Devezas and S. Nunes. "Graph-of-Entity: A Model for Combined Data Representation and Retrieval". In: *Proceedings of the 8th Symposium on Languages, Applications and Technologies (SLATE 2019)*. Vila do Conde, Portugal, 2019. DOI: 10.4230/OASIcs.SLATE.2019.1

J. Devezas and S. Nunes. *Simple English Wikipedia Link Graph with Clickstream Transitions 2018-12 [dataset]*. INESC TEC research data repository. Mar. 2019. DOI: 10.25747/83vk-zt74

**2018**

J. L. Devezas, S. Nunes, A. Guillén, Y. Gutiérrez, and R. Muñoz. "FEUP at TREC 2018 Common Core Track - Reranking for Diversity using Hypergraph-of-Entity and Document Profiling". In: *Proceedings of the Twenty-Seventh Text REtrieval Conference, TREC 2018, Gaithersburg, Maryland, USA, November 14-16, 2018*. Ed. by E. M. Voorhees and A. Ellis. Vol. 500-331. NIST Special Publication. National Institute of Standards and Technology (NIST), 2018. URL: https://trec.nist.gov/pubs/trec27/papers/FEUP-CC.pdf

J. Devezas and S. Nunes. "Social Media and Information Consumption Diversity". In: *Proceedings of the Second International Workshop on Recent Trends in News Information Retrieval co-located with 40th European Conference on Information Retrieval (ECIR 2018), Grenoble, France, March 26, 2018*. 2018, pp. 18–23. URL: http://ceur-ws.org/Vol-2079/paper5.pdf

**2017**

J. L. Devezas, C. T. Lopes, and S. Nunes. "FEUP at TREC 2017 OpenSearch Track Graph-Based Models for Entity-Oriented". In: *Proceedings of The Twenty-Sixth Text REtrieval Conference, TREC 2017, Gaithersburg, Maryland, USA, November 15-17, 2017*. Ed. by E. M. Voorhees and A. Ellis. Vol. 500-324. NIST Special Publication. National Institute of Standards and Technology (NIST), 2017. URL: https://trec.nist.gov/pubs/trec26/papers/FEUP-O.pdf

J. Devezas. *Army ANT: A Workbench for Innovation in Entity-Oriented Search - External Option: Scientific Activities – TREC Open Search*. Research rep. Faculty of Engineering, University of Porto, June 2017. URL: https://hdl.handle.net/10216/110181

J. Devezas. *Auditing Open Access Repositories - Free Option: Supervised Study – Digital Archives and Libraries*. Research rep. Faculty of Engineering, University of Porto, May 2017. URL: https://hdl.handle.net/10216/104152

J. Devezas and S. Nunes. "Information Extraction for Event Ranking". In: *6th Symposium on Languages, Applications and Technologies (SLATE 2017)*. Ed. by R. Queirós, M. Pinto, A. Simões, J. P. Leal, and M. J. Varanda. Vol. 56. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017, 18:1–18:14. ISBN: 978-3-95977-056-9. DOI: 10.4230/OASIcs.SLATE.2017.18

J. Devezas and S. Nunes. "Graph-Based Entity-Oriented Search: Imitating the Human Process of Seeking and Cross Referencing Information". In: *ERCIM News. Special Issue: Digital Humanities* 111 (Oct. 2017), pp. 13–14. URL: https://ercim-news.ercim.eu/en111/special/graph-based-entity-oriented-search-imitating-the-human-process-of-seeking-and-cross-referencing-information

**2016**

T. Devezas, J. Devezas, and S. Nunes. "Exploring a Large News Collection Using Visualization Tools". In: *Proceedings of the First International Workshop on Recent Trends in News Information Retrieval co-located with 38th European Conference on Information Retrieval (ECIR 2016), Padua, Italy, March 20, 2016*. 2016, pp. 48–53. URL: http://ceur-ws.org/Vol-1568/paper9.pdf

J. L. Devezas and S. Nunes. "Index-Based Semantic Tagging for Efficient Query Interpretation". In: *Experimental IR Meets Multilinguality, Multimodality, and Interaction - 7th International Conference of the CLEF Association, CLEF 2016, Évora, Portugal, September 5-8, 2016, Proceedings*. Ed. by N. Fuhr, P. Quaresma, T. Gonçalves, B. Larsen, K. Balog, C. Macdonald, L. Cappellato, and N. Ferro. Vol. 9822. Lecture Notes in Computer Science. Springer, 2016, pp. 208–213. DOI: 10.1007/978-3-319-44564-9_17

# INDEX