Index-based semantic tagging for efficient query interpretation

José Devezas and Sérgio Nunes

INESC TEC & DEI, Faculdade de Engenharia, Universidade do Porto
Rua Dr. Roberto Frias, s/n, 4200-465 Porto, Portugal
{jld,ssn}@fe.up.pt

Abstract. Modern search engines are evolving beyond ad hoc document retrieval. Nowadays, the information needs of the users can be directly satisfied through entity-oriented search, by ranking the entities or attributes that better relate to the query, as opposed to the documents that contain the best matching terms. One of the challenges in entity-oriented search is efficient query interpretation. In particular, the task of semantic tagging, for the identification of entity types in query parts, is central to understanding user intent. We compare two approaches for semantic tagging, within a single domain, one based on a Sesame triple store and another one based on a Lucene index. This provides a segmentation and annotation of the query based on the most probable entity types, leading to query classification and its subsequent interpretation. We evaluate the run time performance for the two strategies and find that there is a statistically significant speedup, of at least four times, for the index-based strategy over the triple store strategy.

Keywords: Entity-oriented search; query segmentation; semantic annotation; query interpretation

1 Introduction

In the last few years, search engines have been evolving from full-text document search into a richer, more entity-oriented search. Entity-oriented or semantic search [2] is a step towards a more direct answer to the user's information needs; it differs from regular full-text search, as results are expected to be entities or attributes, as opposed to full-text search, where results are expected to be documents. Several new problems emerged from the need for entity retrieval. Full-text indexing techniques proved inadequate or insufficient, ranking strategies posed new challenges, as an expanding world of linked data could now contribute to determine the relevance of entities, and the traditional keyword query as a set of terms became unsuitable to support entity-oriented search. When we are looking for entities, we can't necessarily find them through their content, like we do with documents, but rather through their features (e.g., attributes, types, relations). Thus, there is a need to somehow capture and use this information during the search process.

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 [5]. In our system, query interpretation 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 interpretation of the query, helping in the final query answering process.

In this paper, we evaluate the efficiency of the candidate retrieval subtask, based on a Sesame triple store, using SPARQL queries, as well as on a Lucene index, optimized for this task, using keyword queries.

2 Reference Work

Pound et al. [6] 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. [4] 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. [3] 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 49 ms 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 [1] focused on the interpretation 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 *is-a* relations of concepts, while relatedness covered other types of relations).

3 Data Characterization

The work we present here aimed at improving search within the University of Porto. We implemented an entity-oriented search system capable of answering queries by taking advantage of the untapped underlying linked data present in the current information system. We considered search tasks like the discovery of the department for a given staff member or the finding of 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.

The main performance issues that led us to explore an alternative to directly using the triple store for query analysis were identified when we scaled from faculty-centric entities to university-centric entities. 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 triple store 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 more students, associated with 193,650 additional enrollments, 1,166 more courses, 14 more academic years and 10 more faculties.

4 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 triple store 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 triple store 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 will 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 triple store. This contained our knowledge graph and was the obvious choice for an initial approach. As described in Section 3, 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 a Lucene index based on the triple store 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 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.

5 Evaluation

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.

5.1 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 triple store strategy, that we directly compared

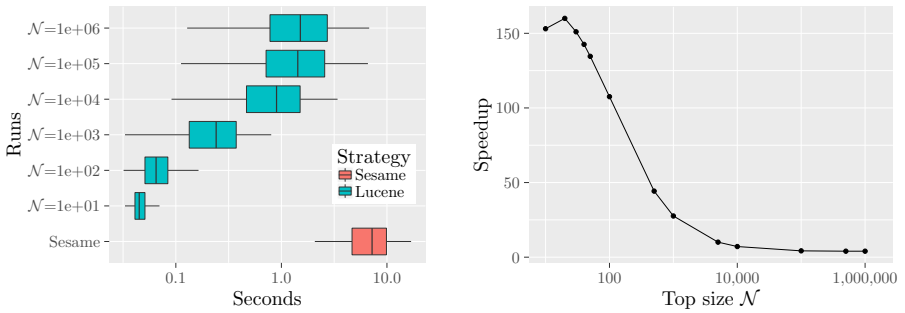
with a run based the Lucene index strategy for the top-$\mathcal{N}$ results. We picked $\mathcal{N} = 10$ since it provided a near-optimal speedup, also having a positive impact on the quality of the results for a small set of manually tested queries.

Table 1: Statistics for the query analysis time of the Sesame triple store and the Lucene index strategies, using $\mathcal{N} = 10$ for the Lucene index.

	Sesame triple store	**Lucene index**
Avg.	7.435765s	0.048580s
Std.	$\pm 3.206806s$	$\pm 0.019115s$
Speedup	153.062268 ($\sim 153\times$ faster)	
Mann-Whitney U Test	p-value $\approx 0 \ll 0.01$	

In Table 1, 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 GB of RAM and a 256 GB solid-state drive. We calculated the speedup of the Lucene index strategy over the Sesame triple store 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 will be shown in Section 5.2.

5.2 Influence of $\mathcal{N}$ over the Speedup



(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).

Fig. 1: Efficiency evaluation of the overall search process. The same 1,000 synthetic queries were used in each run.

Fig. 1a 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 triple store strategy even when retrieving the top $\mathcal{N} = 1$ million matching candidates. Fig. 1b 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 $4\times$ faster.

6 Conclusions

We approached the problem of efficient query interpretation and understanding for entity-oriented search. Based on our practical implementation of a semantic search engine, we proposed a probabilistic methodology for segmenting and annotating query parts with categories, in order to facilitate subsequent interpretation.

We proposed two different strategies for the efficient retrieval of matching candidates from a knowledge base, one of them directly supported on a SPARQL query over a Sesame triple store, and another one supported on a Lucene index directly created from the statements in the knowledge base and built for the specific task of finding candidates that matched different query parts. We evaluated both strategies regarding run time and showed that the index-based strategy outperformed the direct querying of the triple store by a minimum speedup of 4, when retrieving the top 1 million results, and a maximum speedup of over 100, when retrieving a smaller number of results. Based on a synthetic test set of variable length queries, we have shown, with a confidence of over 99%, that the difference in run times between the two strategies was statistically significant.

References

1. Aggarwal, N., Buitelaar, P.: A system description of natural language query over DBpedia. CEUR Workshop Proceedings 913(Ild), 96–99 (2012)
2. Bautin, M.: Entity Oriented Search. Tech. rep., State University of New York at Stony Brook (2007)
3. Blanco, R., Ottaviano, G., Meij, E.: Fast and Space-Efficient Entity Linking for Queries. Proceedings of the Eighth ACM International Conference on Web Search and Data Mining - WSDM '15 pp. 179–188 (2015)
4. Guo, J., Xu, G., Cheng, X., Li, H.: Named entity recognition in query. In: Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2009). pp. 267–274 (2009)
5. Liu, J., Pasupat, P., Wang, Y., Cyphers, S., Glass, J.: Query Understanding Enhanced By Hierarchical Parsing Structures. In: IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU 2013). pp. 72–77 (2013)
6. Pound, J., Mika, P., Zaragoza, H.: Ad-hoc object retrieval in the web of data. In: Proceedings of the 19th international conference on World wide web WWW 10. p. 771 (2010)