Information Extraction for Event Ranking

José Devezas* and Sérgio Nunes
INESC TEC and FEUP InfoLab
jld@fe.up.pt, ssn@fe.up.pt

*Doctoral Program in Computer Science of the Universities of Minho, Aveiro, and Porto (MAP-i)
6th Symposium on Languages, Applications and Technologies, SLATE 2017, June 26-27, ESMAE-IPP, Portugal

A Complete Example: An IE+IR pipeline, from data acquisition to a practical application.

The Big Picture:
It’s time to start fusing techniques in the quest to find the “Master Algorithm”.

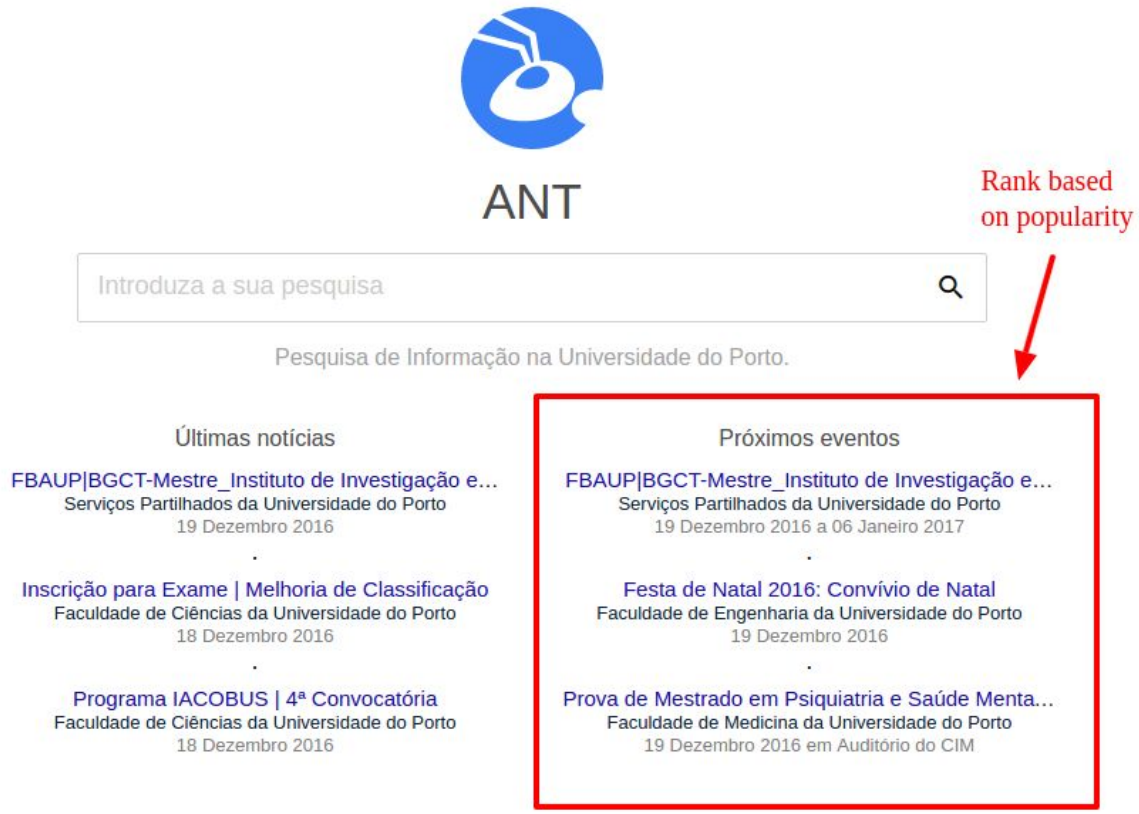
Out-of-the-box thinking can help us define these new research directions.

Bringing research areas like IE and IR closer together will definitely contribute to the effort.

Contents

Target Application

- Query-independent ranking of news that cover event announcements.
- Display the three most relevant upcoming events of general interest to the local academic community.



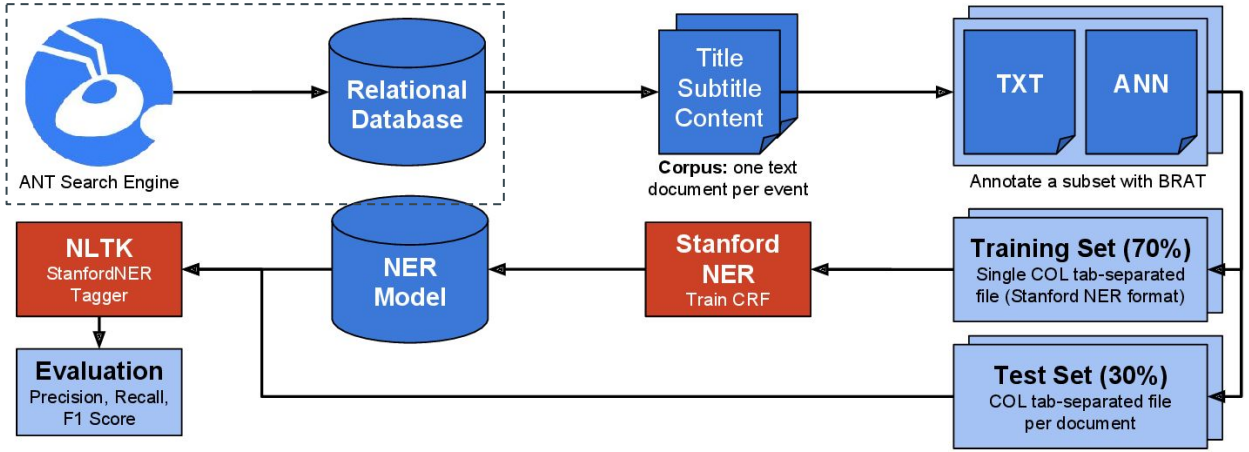
ANT is available at: http://ant.fe.up.pt/

Approach Overview

- Information Extraction pipeline over institutional news:
  - Named Entity Recognition
    - 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*
  - Relation Extraction
    - General event–entity relations;
    - Organization–organization and location–location *partOf* relations.

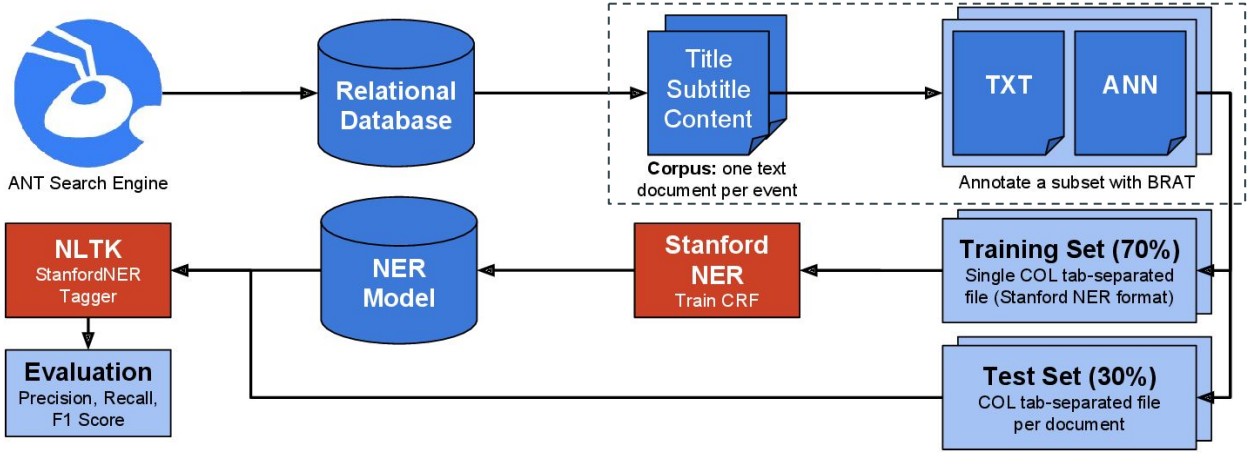
Approach Overview

- Information Extraction pipeline over institutional news (cont.):
  - Knowledge Base construction, mapping identified entities and relations to the following ontologies:
    - Linked Open Descriptions of Events (LODE);
    - DOLCE+DnS Ultralite (DUL);
    - Time Ontology.
- Event ranking based on historical information:
  - News article clicks;
  - Entity popularity:
    - Based on the aggregated number of clicks for news mentioning the entity;
    - Included popularity propagated through *partOf* relations.



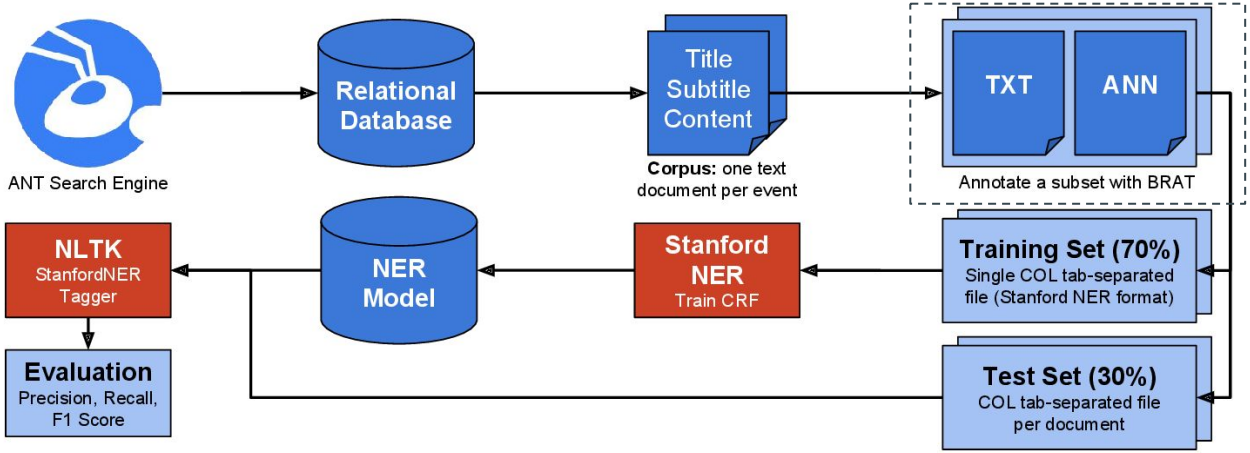
Data Acquisition, Model Training and Evaluation

- Semi-structured data is periodically collected by ANT using XPath and CSS selectors.
  - Student and staff profiles directly represent structured data;
  - But news contain a large textual body of unstructured data.



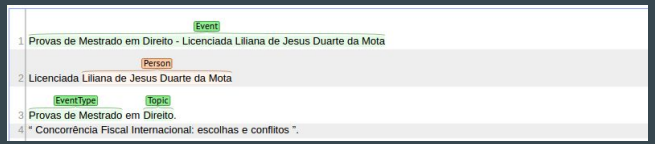
Data Acquisition, Model Training and Evaluation

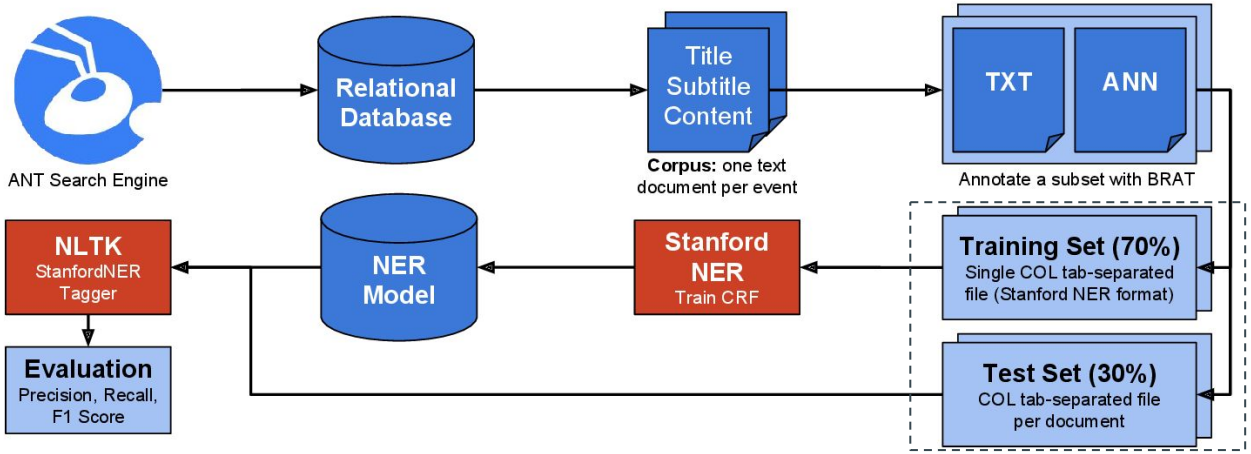
1. Query the relational database for a subset of news articles that announce events and store them as a *CSV* file (one article per row).
2. For each row in the *CSV*, prepare a *TXT* file containing title, subtitle and content, as well as an empty *ANN* file.



Data Acquisition, Model Training and Evaluation

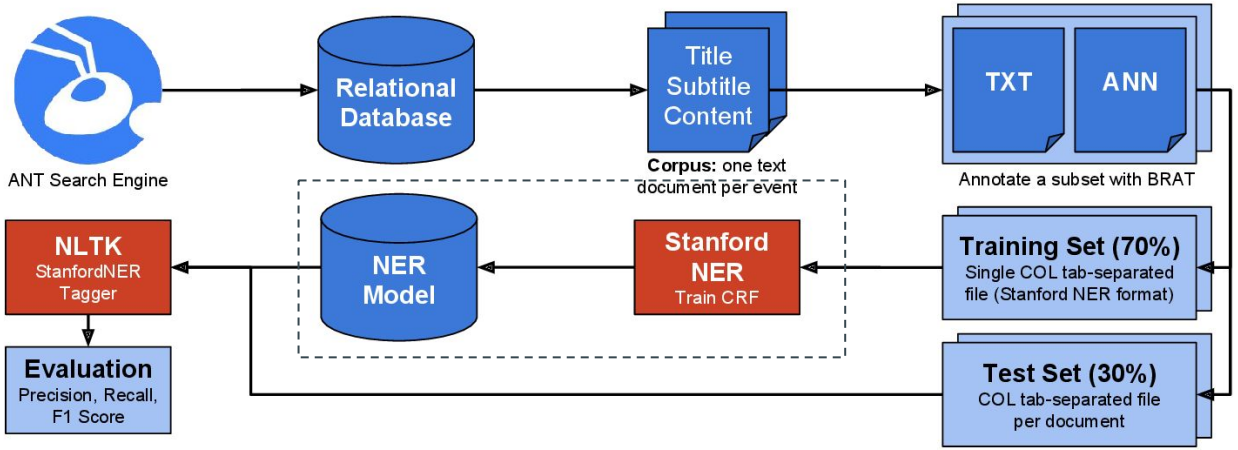
3. Put individual files within a subdirectory in the *data/* directory of the Brat rapid annotation tool and create an annotations.conf file with the list of entity types to annotate.
4. Run *./standalone.py* in Brat's root directory and manually annotate the corpus.





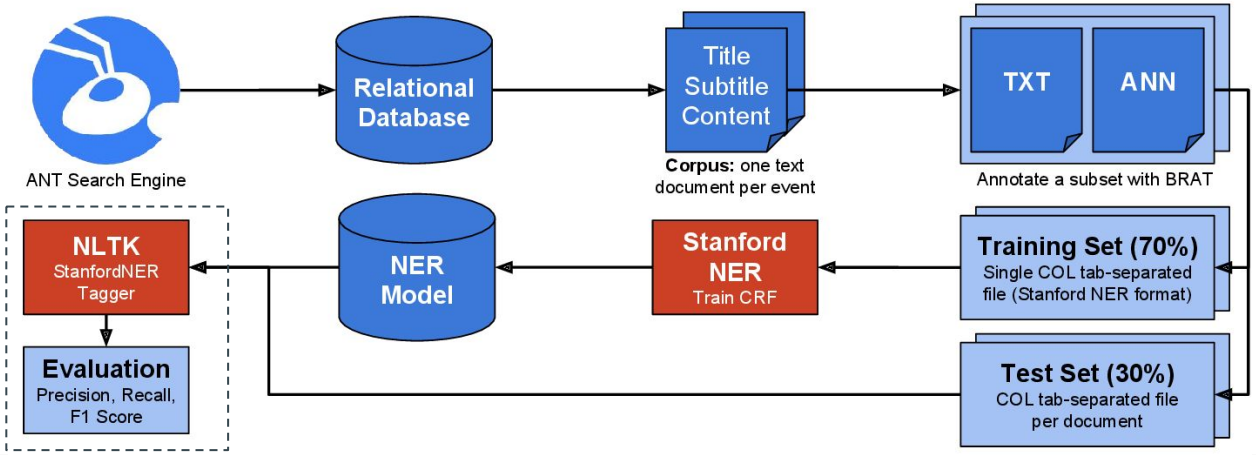
Data Acquisition, Model Training and Evaluation

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).
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.



Data Acquisition, Model Training and Evaluation

7. Train a Conditional Random Field (CRF) using Stanford NER command line interface and obtain a model for named entity recognition (NER).



Data Acquisition, Model Training and Evaluation

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

Avg. Precision: 0.63 | Avg. Recall: 0.37 | Macro F1: 0.47

Information Extraction

NLTK based pipeline:

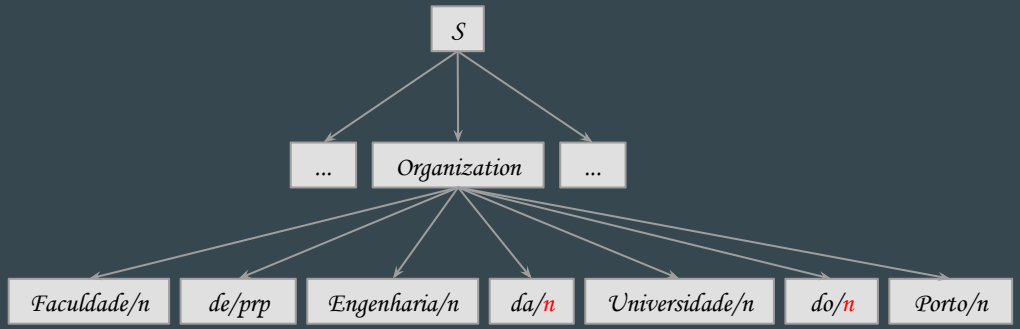
- *detect_language()*
  - Based on *langdetect* Python wrapper;
  - Only ‘pt’ texts are processed.
- *segment_sentences()*
  - Out-of-the-box pre-trained Portuguese model for Punkt sentence tokenizer;
  - Issue: SIGARRA news sometimes contain malformed sentences (e.g., from hypertext lists).
- *tokenize_sentences()*
  - Used the *WordPunctTokenizer* to split each sentence into words and punctuation;
  - This tokenizer implements *span_tokenize()* which was useful to convert Brat’s *ANN* files into Stanford NER *COL* files;
  - Replaced each slash character by a dash, since StanfordNERTagger removed all slashes.

Information Extraction

- *pos_tag_sentences()*
  - Trained with the full Floresta treebank, using backoff tagging:
    - *nltk.BigramTagger* (89% accuracy)
    - *nltk.UnigramTagger* (87% accuracy)
    - *nltk.DefaultTagger* (18% accuracy)
- *ne_tag_sentences()*
  - Assigned a named entity tag to each token using *StanfordNERTagger*;
  - We used the model trained from the annotated SIGARRA News Corpus.

Information Extraction

- *build_sentence_trees()*
  - Build tuples of (word, post_tag, ne_tag);
  - Apply *nltk.chunk.util.conlltags2tree() to* convert to *nltk.tree.Tree* with three levels:
    - Root-level (the sentence);
    - Mid-level (the entity types);
    - Bottom-level (leaves corresponding to chunks of words belonging to a named entity, as aggregated by the mid-level nodes).



Information Extraction

- *extract_entities()*
  - Saves extracted entities to an *ENT* file (custom format for human-readable output) and to a *COL* file for evaluation.
- *extract_relations()*
  - We use *nltk.sem.extract_rels()* to identify relations between two entities, based on a regular expression — we define a list of tuples to iterate over;
    - For example, we define *('Location', '(da|do)/n', 'Location')* to identify *partOf* relations.
  - Each tuple also contains a fourth entry with a list of rules to map the relation to the ontologies.
    - For example, each *Location* is mapped to a *dul:Place* class and the *partOf* relation to a *dul:isPartOf* property.

Information Extraction

- *build_default_relations()*
  - Given the *Event* is the central entity in our system, we can optionally build event–entity relations, which may introduce noise and increasing uncertainty, but also significantly expand the knowledge base;
  - Given our final goal of entity-based event ranking, we include these default relations.
- *load_relations_into_virtuoso()*
  - We generate an N-Triples (*NT*) file with the identified relations for all documents, including *isA* relations (mapped to *rdf:type*);
  - The *NT* file is loaded into OpenLink Virtuoso (our triple store) through a *POST* request to the */sparql-graph-crud-auth* endpoint, storing the information in a separate *ant:EventsKnowledgeBase* graph.

Event Ranking

- Event ranking depends on three factors:
  - Number of days remaining to the event;
  - Entity popularity score;
  - Entity click score.
- We used the SPARQL query to the right to compute the two entity scores: $score_{pop}$ and $score_{clk}$.
  - The statement in orange was removed to compute $score_{pop}$, without the click constraint.
- We only used *dul:Person* and *dul:Organization* entities for this example.

```
SELECT ?event ?code ?school (SUM(?count) AS ?score)
WHERE {
  {
    SELECT ?agent (COUNT(?agent) AS ?count)
    FROM ant:EventsKnowledgeBase
    WHERE {
      ?event a lode:Event .
      ?event ant:wasClicked "true"^^xsd:boolean .
      ?event lode:involvedAgent ?agent .
    }
    GROUP BY ?agent
  }
  ?event a lode:Event .
  ?event lode:involvedAgent ?involved_agent .
  ?agent dul:partOf* ?involved_agent .
  OPTIONAL {
    ?event ant:hasCode ?code .
    ?event ant:hasFaculty ?school .
  }
}
GROUP BY ?event ?code ?school
```

Event Ranking

- The time-to-event factor was combined with the two entity scores as shown in the formula below.
  - We assigned the major weight $w_1 = 0.5$ to the time-to-event factor;
  - Followed by the entity click score with $w_3 = 0.3$;
  - And only then the entity popularity score with $w_2 = 0.2$.

$$score(e, E_e) = w_1 \times \frac{1}{\Delta T_e + 1} + w_2 \times \frac{score_{pop}(e, E_e)}{\max_e\{score_{pop}(e, E_e)\}} + w_3 \times \frac{score_{clk}(e, E_e)}{\max_e\{score_{clk}(e, E_e)\}}$$

Final Remarks

- We presented a simple, yet complete IE pipeline with a practical IR application.
- We have used techniques from two areas (IE+IR), as a first step to start thinking about the unification of existing models.
- We have showcased the power of a knowledge-driven ranking function, through the usage of *partOf* relations to propagate entity popularity.
- We are currently collecting implicit feedback from user clicks, which will enable us to assess the impact of the entity-based event ranking when compared to a basic temporal ranking.

Thanks!

The future of search engine intelligence highly depends on the unified efforts of information extraction and information retrieval. While we already have high quality machine learning techniques to support search by modeling "thought through numbers", we still lack the ability to effectively model "thought through language" in order to build search engines that can better assist users, not only by better understanding them, but also by helping them sort through a large amount of information locked within textual documents in natural language.