ADEL@OKE 2017: A Generic Method for Indexing Knowlege Bases for Entity Linking

Julien Plu[1], Raphaël Troncy[1], Giuseppe Rizzo[2]

[1] EURECOM, Sophia Antipolis, France
julien.plu|raphael.troncy@eurecom.fr
[2] ISMB, Turin, Italy
giuseppe.rizzo@ismb.it

Abstract. In this paper, we report on the participation of ADEL, an adaptive entity recognition and linking framework, to the OKE 2017 challenge. In particular, we propose an hybrid approach that combines various extraction methods to improve the recognition level and an efficient knowledge base indexing process to increase the efficiency of the linking step. We detail how we deal with fine-grained entity types, either generic (e.g. Activity, Competition, Animal for the task 2) or domain specific (e.g. MusicArtist, SignalGroup, MusicalWork for the task 3). We also show how ADEL can flexibly disambiguate entities from different knowledge bases (DBpedia and MusicBrainz). We obtain promising results on the OKE 2017 challenge training dataset for the first three tasks.

Keywords: Entity Recognition, Entity Linking, Feature Extraction, Indexing, OKE Challenge, ADEL

1 Introduction

In this paper, we present our participation to the first three tasks of the OKE 2017 challenge, namely: *1)* Focused NE Identification and Linking; *2)* Broader NE Identification and Linking; *3)* Focused Musical NE Recognition and Linking. This requires to develop a system that can extract a broad range of entity types: generic in the Task 1, fine-grained in the Task 2 or music-specific in the Task 3. This also requires to develop a system that can handle multiple knowledge bases such as DBpedia and MusicBrainz.

We further develop the ADEL framework that is particularly suited to handle such a system adaptivity called by those different requirements [3,4]. We improve the entity extraction and recognition process which now includes a powerful dictionary extractor that handles regular expressions. We also propose a more sophisticated indexing process that allows to index the content of any RDF-based knowledge base such as DBpedia or Musicbrainz. This paper mainly focuses on entity recognition and knowledge base indexing. Entity recognition refers to jointly performing the appropriate extraction and typing of mentions. *Extraction* is the task of spotting mentions that can be entities in the text while *Typing* refers to the task of assigning them a proper type. *Linking* refers to the disambiguation of mentions in a targeted knowledge base. It is also often composed of two subtasks: generating candidates and ranking them accordingly to various scoring functions. Following the challenge requirements, we make use of the 2016-04 snapshot of DBpedia and a 2016-12 snapshot of Musicbrainz as the targeted knowledge bases.

2 Approach

In this section, we describe how we extract mentions from texts that are likely to be selected as entities with the *Extractor Module*. After having identified candidate mentions, we resolve their potential overlaps using the *Overlap Resolution Module*. Then, we describe how we disambiguate candidate entities coming from the extraction step. First, we create an index over a targeted knowledge base, *e.g.* the April 2016 DBpedia snapshot, using the *Indexing Module*. This index is used to select possible candidates with the *Candidate Generation Module*. If no candidates are provided, this entity is passed to the *NIL Clustering Module*, while if candidates are retrieved, they are given to the *Linker Module*.

Extractor Module. We make use of five kinds of extractors: *i)* Dictionary, *ii)* POS Tagger, *iii)* NER, *iv)* Date, and *v)* Number. Each of these extractors run in parallel. At this stage, an entity dictionary reinforces the extraction by bringing a robust spotting for well-known proper nouns or mentions that are too difficult to be extracted for the other extractors. We have developed a new approach for the dictionary extraction that consists in using a generic SPARQL query that will retrieve all entity labels given a list of entity types. We developed a common API for these extractors based on Stanford CoreNLP [2] that is publicly available at `https://github.com/jplu/stanfordNLPRESTAPI`.

Indexing Module. An index can be seen as a two-dimensional array where each row is an entity in the index and each column is a property that describes the entity. Indexing the English DBpedia snapshot and retaining only properties that have literal values yields 281 columns. Once we have this index, we can search for a mention in this index and retrieve entity candidates. Searching, by default, over all columns (or properties used in the knowledge base), negatively impacts the performance of the index in terms of computing time. In order to optimize the index, we have developed a method that maximizes the coverage of the index while querying a minimum number of columns (or properties) [5]. For the DBpedia version 2016-04, there are exactly 281 properties that have literal values, while our optimization produced a reduced list of 8 properties: *dbo:wikiPageWikiLinkText*, *dbo:wikiPageRedirects*, *dbo:demonym*, *dbo:wikiPageDisambiguates*, *dbo:birthName*, *dbo:alias*, *dbo:abstract* and *rdfs:label*. This optimization drastically reduces the time of the query from around 4 seconds to less than one second. The source code of this optimization is also available[3]. Previously, we were using an index stored in Lucene. We have, however, observed unexpected behavior from Lucene such as not retrieving resources that partially match a query even if the number of results was not bound due to the lack of parameters and control of what can be searched on. The index is now built using *Elasticsearch* as a search engine which provides better scoring results. The indexing of a knowledge base follows a two-step process: *i)* extracts the content of a knowledge base, and creates the Elasticsearch index; *ii)* runs the optimization method in order to get the list of columns that will be used to query the index.

NIL Clustering Module. We propose to group the *NIL* entities (emerging entities) that may identify the same real-world thing. The role of this module is to attach the

[3] `https://gist.github.com/jplu/a16103f655115728cc9dcff1a3a57682`

same *NIL* value within and across documents. For example, if we take two different documents that share the same emerging entity, this entity will be linked to the same *NIL* value. We can then imagine different *NIL* values, such as *NIL_1*, *NIL_2*, etc. We perform a string strict matching over each possible NIL entities (or between each token if it is a multiple token mention). For example, in that Task 1 dataset, sentence 23, both the mention "Sully" and the mention "Marine Jake Sully" will be linked to the same NIL entity. We are also testing a version that uses co-references [5].

Linker Module. This module implements an empirically assessed function that ranks all possible candidates given by the *Candidate Generation Module*:

$$r(l) = (a \cdot L(m, title) + b \cdot max(L(m, R)) + c \cdot max(L(m, D))) \cdot PR(l) \qquad (1)$$

The function $r(l)$ is using the Levenshtein distance L between the mention m and the title, and optionally, the maximum distance between the mention m and every element (title) in the set of Wikipedia redirect pages R and the maximum distance between the mention m and every element (title) in the set of Wikipedia disambiguation pages D, weighted by the PageRank PR, for every entity candidate l. The weights a, b and c are a convex combination that must satisfy: $a + b + c = 1$ and $a > b > c > 0$. For our experiments, the values of those parameters are $a = 16/21$; $b = 4/21$ and $c = 1/21$ as defined in [3,4]. We take the assumption that the string distance measure between a mention and a title is more important than the distance measure with a redirect page which is itself more important than the distance measure with a disambiguation page. Contrarily to DBpedia, we do not have a computed PageRank for the MusicBrainz entities. For this reason, we replace the PageRank score by a scoring function that modify the score retrieved by Elasticsearch[4] in normalizing it between 0 and 1, while redirect and disambiguation pages are ignored in the case of MusicBrainz.

3 Preliminary Results and Discussion

We present some preliminary results for the Task 1 (Table 1) and Task 2 (Table 2), performing a four cross-fold validation using the given training set for each task provided the size of the OKE 2017 dataset that prevents to use more folds. The three default models from Stanford NER (3-classes, 4-classes and 7-classes) trained on the CoNLL 2003 dataset have been combined with the NER model trained on each fold, in this order, using the NER model combination algorithm described in [4].

No other extractors have been used. We have conducted multiple other experiments using the flexibility offered by ADEL and we report below some lessons learned.

ADEL offers a broad range of entity extractors. In the Task 1, we have tried to include the Dictionary extractor. This leads to a precision decrease of 13% with a gain of 2% in recall. We also tried to include the POS extractor that considers that any singular or plural proper nouns are mentions. This leads to a precision decrease of 36% with a gain of 5% in recall. Consequently, we observe that while those two extractors enable

[4] `https://www.elastic.co/guide/en/elasticsearch/reference/current/search-request-rescore.html`

	Precision	**Recall**	**F1**
strong_mention_match	74.9	75.4	75.1
strong_link_match	47	54.5	50.3

Table 1: Results for Task 1 using the nel-eval scorer

	Precision	**Recall**	**F1**
strong_mention_match	70.7	67.9	69.2
strong_link_match	40.8	45.8	42.9

Table 2: Results for Task 2 using the nel-eval scorer

to find more mentions, they also bring significant noise. In the Task 1 dataset, the dictionary contains 423.513 entries while for the Task 2 dataset, 2.094.719 entries have been added. Therefore, dictionaries bring back the issue of dealing with synonymy. For example, in Task 1, the sentence 22 contains the preposition *against* while a few bands named *Against* exist in the knowledge base[5]. The Dictionary will extract any occurrence of the preposition mixing it with the band. The POS extractor will extract all proper nouns as mentions, even though those were not annotated in the Task dataset. For example, in Task 1 in the same sentence 22, the mention *World War II* will be extracted, but this entity does not belong to the types that are considered in the task definition.

We made the same observations experimenting on the dataset of Task 2. Adding the Dictionary extractor leads to a precision decrease of 15% with a gain of 4% in recall. Adding the POS extractor leads to a precision decrease of 24% with a gain of 8% in recall. In terms of F1, adding those two extractors penalize less in Task 2 than in Task 1 since a much broader set of entity types should be extracted in the Task 2.

We observe a significant drop in performance at the disambiguation stage. The linking formula which is currently being used is sensitive to the noise brought at the extraction step since this module does not take into account the entity context but instead relies on a combination of string distances and the PageRank global score. For example, in the Task 1 dataset, sentence 1, the string distance score over the title, the redirect and the disambiguation pages between the mention *Trump* and the entity candidate `db:Trumpet` is higher than with the correct entity candidate `db:Donald_Trump`.

We also evaluate the efficiency of our candidate generation module that, given a mention, should always provide the correct disambiguation link among a set of candidates. The evaluation is done as follows: from a training dataset, we perform a SPARQL query in order to get all mentions with their disambiguation link; then, for each mention, we query our index by using the list of columns listed in Section 2 to get a set of candidates and we check if the proper link is contained in that set. The minimum index of the correct link in this set is 1 while the maximum index is 1729 for the Task 1 and 1943 for the Task 2. For the Task 1, the recall@1729 is 94.65% and for the Task 2, the recall@1943 is 90.22%. Most often, when the correct link is not retrieved, it is because the mention does not appear in the content of the queried columns, such as *007's*[6] in the sentence 37 of the Task 1 dataset.

[5] `https://en.wikipedia.org/wiki/Against`

[6] `http://dbpedia.org/resource/James_Bond_(literary_character)`

4 Conclusion and Future Work

We have presented an entity extraction and linking framework that can be adapted to the kind of entities that have to be extracted and adapted to the knowledge base to disambiguate to making it particularly suited for participating in 3 tasks of the OKE 2017 challenge. While the recognition and the candidate generation process provide good performance, the linking step is currently the main bottleneck in our approach. The performance drops significantly at this stage mainly due to a fully unsupervised approach.

We plan to investigate a new method that would modify Deep Structured Semantic Models [1] to make it compliant with knowledge bases and use it as a relatedness score between each candidate to build a graph composed of these candidates where each edge is weighted by this score. The path that has the highest score is chosen as the good one to disambiguate each extracted entity. This method should be agnostic to any knowledge base as it will uses the relations among the entities. We also plan to align the entity types from different NER models, in order to have a more robust recognition step. The association of multiples kind of extraction techniques makes our approach extracting a significant amount of false positives. For this reason, we are also investigating to add a pruning step at the end of the process in order to reduce this amount of false positives. Finally, to improve the extraction by dictionary, we plan to make an automated regular expression generator that, given an entity, would match as many cases as possible. SPARQL queries using those seeds will then generate a dictionary composed of regular expressions that would match multiple derivation of the entities.

Acknowledgments

This work was primarily supported by the innovation activity PasTime (17164) of EIT Digital (`https://www.eitdigital.eu`).

References

1. P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck. Learning Deep Structured Semantic Models for Web Search Using Clickthrough Data. In *22nd ACM International Conference on Information & Knowledge Management (CIKM)*, 2013.
2. C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. The Stanford CoreNLP Natural Language Processing Toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, 2014.
3. J. Plu, G. Rizzo, and R. Troncy. A Hybrid Approach for Entity Recognition and Linking. In *12th European Semantic Web Conference (ESWC), Open Knowledge Extraction Challenge*, 2015.
4. J. Plu, G. Rizzo, and R. Troncy. Enhancing Entity Linking by Combining NER Models. In *13th European Semantic Web Conference (ESWC), Open Knowledge Extraction Challenge*, 2016.
5. J. Plu, G. Rizzo, and R. Troncy. ADEL: ADaptable Entity Linking. *Semantic Web Journal (SWJ), Special Issue on Linked Data for Information Extraction*, (under review), 2017.
6. G. Rizzo, M. van Erp, and R. Troncy. Inductive Entity Typing Alignment. In *2nd International Workshop on Linked Data for Information Extraction (LD4IE)*, 2014.