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 the participation of ADEL to the OKE 2017 challenge. In particular, an adaptive entity recognition and linking framework that combines various extraction methods for improving the recognition level and implements an efficient knowledge base indexing process to increase the performance of the linking step. We detail how we deal with fine-grained entity types, either generic (e.g. Activity, Competition, Animal for Task 2) or domain specific (e.g. MusicArtist, SignalGroup, MusicalWork for Task 3). We also show how ADEL can flexibly link entities from different knowledge bases (DBpedia and MusicBrainz). We obtain promising results on the OKE 2017 challenge test 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. The participation to these tasks has required to develop a system that can extract a broad range of entity types: generic in the Task 1, fine-grained in Task 2 or music-specific in Task 3. This has also triggered to develop a system that can handle multiple knowledge bases, such as DBpedia and MusicBrainz, to link the spotted candidates to referent resources.

We further develop the ADEL framework that is particularly suited to be adaptable to each of the requirements [3,4].

We improve the entity extraction and recognition process that includes a 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 or link them to NIL if no candidates are found. 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.

The rest of the paper is structured as follows: Sections 2 introduce our approach, Section 3 proposes the evaluations of this approach over each test dataset of OKE2017. Finally, conclusions and future work are discussed in Section 4.

2 Approach

In this section, we describe how we extract mentions from texts that are likely to be selected as entities by 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 the English DBpedia snapshot (version 2016-04) 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 retrieves 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 queryig by a factor of 4, in detail from 4 seconds to less than one second on a server that has 256GB of RAM and a Intel Xeon CPU E5-2670 v3 @ 2.30GHz. The source code of this optimization is also available[3].

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

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 that 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 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 sentence 23 of the datased used for Task 1, both the mention "Sully" and "Marine Jake Sully" will be linked to the same NIL entity.

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

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

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 $a > b > c > 0$. 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 that is itself more important than the distance measure with a disambiguation page. In DBpedia not all pages have redirect or disambiguation pages associated, for this reason the two last elements of the formula are optional. This means that if a page does not have redirect pages, only the title and the disambiguation pages are evaluated, and the same logic is applied when only disambiguation pages exist, and finally, if no redirect and disambiguation pages exist, only the title is taken into account. In order to also apply this formula with Musicbrainz entities, we have computed a PageRank for each of them.

3 Results and Discussion

In the OKE2107 challenge the evaluation for each task had two different scenarios: *A)* the goal is to evaluate the performance of the linking by achieving the highest F1-score, *B)* the goal is to evaluate the best ratio $\beta = \frac{F1-score}{runtime}$ of the system. The official OKE 2017 released scores for Scenario A are reported in Table 1. For Scenario B, the results are reported in Table 2. The comparison in each table is done with FOX [7,8] which was the baseline for each task. We have used the same ADEL configuration for each task:

1. Extraction: three different extractors, *i)* Stanford NER with the 3-class, 4-class and 7-class models, *ii)* Stanford NER with the model trained with the training set of the corresponding task, and *iii)* a specific gazetteer made for the corresponding task.
2. Index: we use the DBpedia index for the two first tasks, and the Musicbrainz one for the third task. The Elasticsearch query we used to get the candidate has been adapted for each task.
3. Linking: we used the same weights for all tasks: $a = \frac{16}{21}$, $b = \frac{4}{21}$, and $c = \frac{1}{21}$.

This ultimately generated three different ADEL instances, one for each task.

		ADEL			FOX		
		Precision	Recall	F1	Precision	Recall	F1
Task 1	Recognition	91.62	83.20	87.21	92.47	80.58	86.12
	D2KB	40.15	27.82	32.87	61.96	41.47	49.69
	A2KB	33.24	30.18	31.64	53.61	46.72	49.93
Task 2	Recognition	87.68	78.57	82.88	95.9	65.80	78.05
	D2KB	39.93	25.75	31.32	63.42	35.28	45.34
	A2KB	31.4	28.14	29.68	56.15	38.53	45.7
Task 3	Recognition	35.03	74.57	47.66	63.02	49.21	55.27
	Typing	64.33	64.91	64.62	0	0	0
	RT2KB	26.99	27.24	27.12	0	0	0

Table 1: Results for Scenario A over the OKE2017 datasets for the three tasks.

Concerning Task 1 and Task 2, the first thing we observe is the efficiency of the extraction part in ADEL, that of course can be leveraged depending the combination of extractors we use.

We can also observe that this extraction depends of what we want to extract, the more complex are the types to extract the more difficult is the extraction, and ADEL is robust against that, because despite the different number of entity types that must be extracted in Task 1 and Task 2, the F1 score shows a small difference between the two tasks and then proves ADEL robustness compared to FOX.

Task 3 provides fine grained and specific entity types (artists, songs and albums), which bring a major issue: the name of an artist, a song or an album can be anything, including, for instance, a punctuation mark[4], for this reason we have preferred to configure ADEL to have a high recall.

For all tasks we observe a significant drop in performance at the linking stage. The linking formula 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 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 the correct entity candidate `db:Donald_Trump`.

[4] https://musicbrainz.org/work/25effd3c-aada-44d1-bcbf-ede30ef34cc0

		ADEL			FOX		
		β	F1-score points	avg Millis Per Doc	β	F1-score points	avg Millis Per Doc
Task 1	overall	0.0009	114.58	231314.48	0.0036	363.25	179287.18
	1	0.045	16.42	4613.25	0.024	50.17	26612.1
	2	0.01	16.49	20851.94	0.027	55	25808.69
	3	0.003	14.39	60331.46	0.011	55.43	63613.36
	4	0.00095	13.81	182078.99	0.0043	50.088	146824.36
	5	0.00074	19.65	337788.62	0.0028	53.63	240083.76
	6	0.00047	17.52	462000.34	0.0019	50.11	330158.19
	7	0.00038	16.3	567022.28	0.0015	48.82	420001.39
Task 2	overall	0.00078	102.48	261497.17	0.0015	208.85	245985.25
	1	0.032	15.30	5849.78	0.01	29.28	35759.16
	2	0.011	14.9	17087.79	0.0085	26.71	39090.31
	3	0.002	15.26	93618.21	0.004	34.96	109790.31
	4	0.00088	18.17	258233.61	0.0014	26.72	237480.25
	5	0.00059	16.71	399959.27	0.0011	30.63	347780.25
	6	0.00041	12.06	538963	0.00092	31.76	433000.36
	7	0.00023	10.086	746879.68	0.0007	28.8	518996.09

Table 2: Results for Scenario B over the OKE2017 datasets for the two first tasks.

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 Task 1, 1943 for Task 2, and 673 for Task 3. For Task 1 the recall@1729 is 94.65%, for Task 2 the recall@1943 is 90.22%, and for Task 3 the recall@673 is 97.32%. 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*[5] in the sentence 37 of Task 1 dataset.

Regarding Scenario B in Table 2, we can see that ADEL has a drop of performance in terms of average millis per document from the 4th phase. In order to understand why this drop, we have profiled ADEL to detect the possible bottlenecks using the test dataset of Task 1. All the identified bottlenecks here are mostly observations that affect the runtime performance of ADEL. We succeeded to identify two significant bottlenecks: *i)* the network latency, and *2)* the candidate generation. The first is due to a high usage of external systems via HTTP queries (all the extractors and Elasticsearch), the sum of the latency of each HTTP query penalizes the runtime of ADEL. Unfortunately, we cannot really do something to solve this as it is an ADEL requirement to use external systems. Finally, the second bottleneck is Elasticsearch, arriving to a certain number of queries our ADEL instance gets stuck and starts to queue the queries. To solve this problem, we have developed a new architecture for our cluster by making each node

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

able to be queried via a load balancer system. This solution allows to increase the number of queries run in the same time without being queued. This new architecture has divided the time to get our candidates by almost three (approximately one division per node).

4 Conclusion and Future Work

We have presented an entity extraction and linking framework that can be adapted to the entity types that have to be extracted and adapted to the knowledge base used to link the spotted entities. We have applied this framework to 3 tasks of the OKE 2017 challenge. While both recognition and the candidate generation processes 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 use the relations among the entities. We also plan to align the entity types from different NER models, exploiting and extending previous work [6], in order to have a more robust recognition step. The association of multiples types 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 the amount of false positives. Finally, to improve the extraction by dictionary, we plan to make an automated regular expression generator that, given an entity, will 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 *22^{nd} 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 *12^{th} 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 *13^{th} 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 *2^{nd} International Workshop on Linked Data for Information Extraction (LD4IE)*, 2014.
7. R. Speck and A. Ngomo.
8. R. Usbeck, A. Ngomo, M. Röder, D. Gerber, S. Coelho, S. Auer, and A. Both. Agdistis - graph-based disambiguation of named entities using linked data. In *The Semantic Web – ISWC 2014*, 2014.