UKP-UBC Entity Linking at TAC-KBP

Nicolai Erbs[†], Eneko Agirre[‡], Aitor Soroa[‡], Ander Barrena[‡], Ugaitz Etxebarria[‡], Iryna Gurevych[†], Torsten Zesch[†]

[†]Ubiquitous Knowledge Processing Lab (UKP-TUDA)
Department of Computer Science, Technische Universität Darmstadt
http://www.ukp.tu-darmstadt.de

[‡]IXA NLP Group
University of the Basque Country, Donostia, Basque Country
http://ixa.si.ehu.es

Abstract

This paper describes our system for the entity linking task at TAC KBP 2012. We developed a supervised system using dictionary-based, similarity-based, and graph-based features. As a global feature, we apply Personalized PageRank with Wikipedia to weight the list of entity candidates. We use two Wikipedia versions with different timestamps to enrich the knowledge base and develop an algorithm for mapping between the two Wikipedia versions. We observed a large drop in system performance when moving from training data to test data. Our error analysis showed that the guidelines for mention annotation were not followed by annotators. An additional mention detection component should improve performance to the expected level.

1 Introduction

Entity linking is the task of linking a surface string in a text (e.g. *Washington*) to the correctly disambiguated representation of the entity in some knowledge base (e.g. either *George Washington* or *Washington, D.C.* depending on what *Washington* was referring to in this context). Entity linking is an important task with applications in several domains including information retrieval, text structuring, and machine translation. The Knowledge Base Population workshop of the Text Analysis Conference[1] provides a framework for comparing different approaches to entity linking in a controlled setting. We focus on the entity linking task which assigns an entity from a knowledge base to a marked mention in a text.

Our approach is solely based on the given mention (no further approaches for extending the mention are used) and ranking the candidate entities. This enables the system to be used for rather general problems like word sense disambiguation (Agirre and Edmonds, 2006; Navigli, 2009).

This paper is structured as follows: Section 2 provides an overview of the notation used throughout this paper. In Section 3, we describe the system architecture and classify the features used in Section 4. We present the official results and our error analysis in Section 5 and conclude with a description of future work (Section 6) and a summary (Section 7).

2 Notation

Each document d contains exactly one mention m with a context c. For each mention m, there exists a set of entities E, which are entity candidates for m. Each feature has a scoring function $s(m, e)$ that computes the probability for each mention-entity pair (m, e) that m refers to e. For features that consider the context the scoring function $s(m, e, c)$ also takes the context into account.

3 System architecture

Figure 1 shows the architecture of the system. The system is designed using modular components based on the UIMA framework (Ferrucci and Lally, 2004) augmented with the functionality provided by uimaFIT (Ogren and Bethard, 2009).

[1] http://www.nist.gov/tac/2012/KBP/index.html

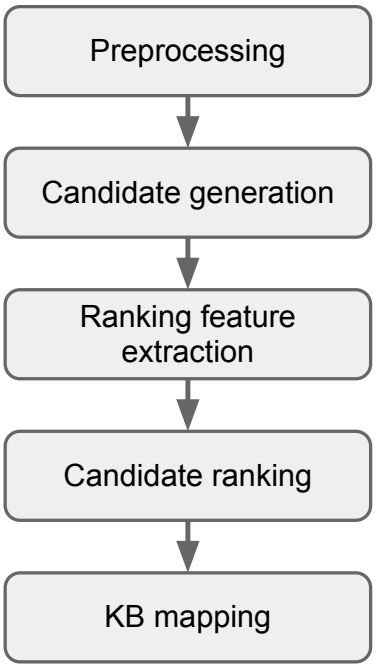

Figure 1: System architecture

Preprocessing Preprocessing components were taken from the open source project DKPro Core[2]. We used tokenization, POS-tagging, lemmatization, chunking, and named entity recognition.

Candidate generation We reduce the whole set of entities E to those that have previously been used as a target for the corresponding mention m. For this purpose we use two types of dictionaries: One is extracted from existing links in Wikipedia (Chang et al., 2010), redirects, and category information. The other is extracted from Google search logs (Spitkovsky and Chang, 2012) and incorporates additional information like source language. A Wikipedia dictionary is specific for one Wikipedia timestamp while information in the Google dictionary is collected over a 2-year period.

The system uses several dictionaries of those two types in parallel to overcome the disadvantages of each dictionary. A dictionary constructed using a Wikipedia version close to the KB timestamp (i.e. 12th of March 2008) covers all entities in the KB but contains fewer mention-entity pairs, due to the smaller size of Wikipedia in 2008. A dictionary constructed from a more recent Wikipedia (i.e. 5th of April 2011) provides a fair number of mention-entity pairs, but may contain entities that have been renamed, merged, split, or deleted since the KB-version in 2008. The Google dictionary provides statistical information for basically any mention, but is sometimes noisy.

Candidate expansion When the target query string is not found in the dictionary, the following heuristics are applied in turn to the mention:

1. remove parentheses and text in between
2. remove leading "the" from mention

In one run that accessed external resources queried online, we also use the Did You Mean (DYM) API in Wikipedia. First, we check if the dictionary lookup returns any entity candidates. If not, we try DYM. If DYM fails to return any candidates, it is applied after each of the above heuristic again. That is, if the first heuristic fails, then we try DYM for the mention without parenthesis, and so on.

Ranking feature extraction For each mention-entity pair (m, e), each of the features computes a scoring function $s(m, e)$ or $s(m, e, c)$ if the context is considered. Details of these features are described in Section 4.

Candidate ranking In order to rank entity candidates, several options are available:

- the score of a single feature
- a linear combination of many or all features
- a supervised component that learns a model

We submitted several runs for different feature combinations. The supervised combinations were generated with Rank SVM (Tsochantaridis, 2006).

KB mapping Due to constant revisions of Wikipedia articles, not all articles can be mapped to older versions using their title. Hence, we follow a three-step process: First, we try to map an article by its title. Second, we search for a redirect from the corresponding Wikipedia version with the same title and map it to its target. Third, heuristics as previously described for candidate expansion are used to map entities.If none of these methods return a KB entry, NIL (Not in KB) is returned.

[2] code.google.com/p/dkpro-core-asl/ and code.google.com/p/dkpro-core-gpl/

4 Classification of features

In this section, we give a brief overview of the feature we used in our entity linking system. We divide into three types of features: (i) features solely considering the mention, (ii) features taking the context into account, and (iii) another contextual feature doing a global disambiguation.

4.1 Mention

These features take the mention and a list of entity candidates as input.

Dictionary-based. This feature computes a score for any tuple (m, e) derived from the frequency of a dictionary including target statistics for each mention. Two different types of dictionaries from distinct sources are used:

- *Wikipedia* (Chang et al., 2010). A Wikipedia-based dictionary that lists all mentions with a frequency distribution of their target. It is generated by adding all categories, redirects, and all links in all Wikipedia articles of one language version. Statistical data slightly differs depending on the time stamp of the Wikipedia version. Probability may change over time since Wikipedia article are under constant revision, e.g. they are added, split, merged, or removed. Hence, different Wikipedia version may return different frequency distributions. In our system we use two previously mentioned Wikipedia versions, one from 2008 and one from 2011.

- *Google* (Spitkovsky and Chang, 2012). This resource collects information from Google search logs and therefore is able to provide statistics from a large crowd of users. It also includes information about Wikipedia inter-language links, and types (e.g. disambiguation pages) of the entity page in Wikipedia.

Similarity-based. The similarity-based feature computes the score $s(m, e)$ according to the string similarity between the mention and the title of the candidate entity. This is useful for misspellings, e.g. the wrongly written name *Georg Washington* is much closer to *George Washington* than to *Washington D.C.*. We use the open-source package DKPro Similarity[3] and apply Levenshtein and Jaro-Winkler distance.

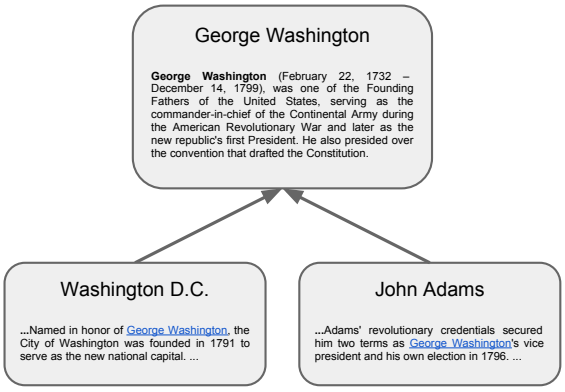

Figure 2: Building description text for *George Washington* using text from its article and articles that refer to that page.

4.2 Context

A common approach in Word Sense Disambiguation is to use the context of the mention to identify the correct sense. For instance, in the Lesk algorithm (Lesk, 1986) words in the context of the mention are compared to words in the description of each entity given by its Wikipedia article. We use a similar approach, also used by Han and Sun (2011), where context words are weighted using the frequency in the description, smoothed by their n-gram frequency (Jelinek and Mercer, 1980). That way, high-frequent words have a lower influence than uncommon words, similar to tf.idf weighting (Salton and Buckley, 1988). We use the Google Web 1T corpus (T. Brants and Franz, 2006) for frequency counts.

As an alternative to the use of the description in the corresponding Wikipedia article, we extract the link context of every incoming link to the corresponding page. Figure 2 shows an example for *George Washington* which is – among many others – linked from *Washington D.C.* and *John Adams*. Either the document text of the article can be used or the context of the link anchors pointing to the article for *George Washington*.

[3] code.google.com/p/dkpro-similarity-asl/

4.3 Global

These features use the context of the mention and disambiguate the marked mention and all further mentions in the context, collectively, seeking to optimize the relation between the entities. For each mention in the context and the marked mention itself, every possible candidate is mapped to a Wikipedia article. The context is transformed from a list of words to lists of Wikipedia articles.

Wikipedia incoming links As a baseline approach, we use the number of incoming links shared between two articles in Wikipedia. This approach is similar to the relatedness measure by Milne and Witten (2008).

Personalized PageRank The PageRank random walk algorithm (Brin and Page, 1998) is a method for ranking the vertices in a graph according to their relative structural importance. PageRank starts by taking a graph $G = (V, E)$ with a set of vertices V representing concepts and a set of edges E representing relations between them. PageRank can be viewed as random walk processes that postulate the existence of a particle that randomly traverses the graph, but at any time may jump, or *teleport*, to a new vertex with a given *teleport probability*. In standard PageRank this target is chosen uniformly, whereas for the so called *Personalized PageRank* (Haveliwala, 2002), the teleport probability is chosen from a nonuniform distribution of nodes, specified by a *teleport vector*.

The final weight of node i represents the proportion of time the random particle spends visiting it after a sufficiently long time, and corresponds to that node's structural importance in the graph. Because the resulting vector is the stationary distribution of a Markov chain, it is unique for a particular walk formulation. As the teleport vector is nonuniform, the stationary distribution will be biased towards specific parts of the graph.

In our system we use a dump of Wikipedia as a graph. Specifically, Wikipedia articles are the graph vertices, and we link two vertices a_1 and a_2 whenever there exist links in both directions between articles a_1 and a_2 in Wikipedia (i.e., links from a_1 to a_2 and from a_2 to a_1). The resulting graph contains $2,325,876$ vertices and $5,549,696$ edges.

Run	System	Training	Test
#1	RankSvm (selected features)	.788	**.316**
#7	Context with entity title	.777	**.377**

Table 1: B^3+ F1 (All) results for train and test data

For each document to be processed, the teleport vector is initialized as follows: let $\{m_1, \ldots, m_N\}$ be the possible mentions found in the document, and let $A = \{a_1, \ldots, a_M\}$ be all the possible candidates for the mentions $m_1, \ldots, m_N$. Note that the query string is included in the mention set.

We initialize the teleport vector by assigning some value to the vertices a_i and zero on the rest. We experimented with two approaches to assign values to the articles:

- The weight approach: we initialize each article with $s(m, e)$, the probability of the particular article given the mention.
- The no weight approach: we initialize each article in A with 1.

We normalize the teleport vector so that its elements sum to one and apply Personalized PageRank using the UKB package[4]. After Personalized PageRank computation, we output the final ranks of the articles which are the possible candidates of the query string. These ranks are used as input features for the classifier described in Section 3.

The PageRank algorithm needs some parameters to be specified, namely the so-called damping factor and the convergence threshold (Brin and Page, 1998). Following usual practice, we used a damping value of 0.85 and finish the calculations after 30 iterations (Agirre and Soroa, 2009). We did not tune these parameters.

5 Results and error analysis

Table 1 shows our results for the training data[5] and official results for the test data. The evaluation metric used is B^3+ F1 on the complete data set.B^3+ takes the correctly identified entities into account and in case mentions should be linked to a NIL

[4] `http://http://ixa2.si.ehu.es/ukb/`

[5] Previous entity linking data from 2009 to 2011

entity[6] it also checks the correct clustering of all NIL mentions. We show evaluation results for the best performing systems on training and test data. The best performing system on training data is the supervised system using selected features[7] (cross-validation is applied). The best performing system on test data is a context feature using only the entity candidate title.

The results show that our test results are drastically lower than our results on the training data. We analyzed the poor performance on the test data and detected a problem in the candidate identification step.

The official TAC guidelines used by the annotators to develop the test queries states in Section `2.3 Complete Mentions`[8]:

> Name strings that are substrings of a complete named-entity mention are not allowed for any queries. For example, given the complete name string "John Smith" in a document, you cannot select either the words "John" or "Smith" by themselves as they constitute a substring of a full mention. Additionally, you should not select "Springfield" from the sentence "She grew up in Springfield, OH". These words could only be selected if they appeared separately in the document.

We identified a large number of cases not following this guideline, e.g. *Collins* instead of *Gary Collins* and *New Kids* instead of *New Kids on The Block*. In the case of *Collins*, our system generated many results which made it very difficult for the ranking. In the case of *New Kids*, our system did not generate any entity candidates.

The large number of cases with incomplete mentions is the major reason for the poor performance on test data. This shifts the task of entity linking from a ranking task to a mention identification task: If the marked mention is used without further preprocessing, candidate generation and ranking is excessively difficult.

[6] Entities not in the knowledge base

[7] UKB, Google dictionary, context-based with titles

[8] `http://www.nist.gov/tac/2012/KBP/task_guidelines/TAC_KBP_Entity_Selection_V1.1.pdf`

6 Future work

In the future, we plan to include mention identification components, i.e. methods identifying the complete mention in an input document. This also includes methods for acronym expansion and improved named entity recognition. We will also focus on advanced methods for global disambiguation like graph algorithms and will include richer linguistic information extracted from UBY (Gurevych et al., 2012).

7 Summary

We presented our contribution to the TAC KBP 2012 entity linking task. We described the architecture of our supervised system and classified the features according to their input parameters. Our entity linking framework is expandable and based on open-source software libraries. Due to truncated mentions in the test data and resulting violation of the annotation guidelines of the challenge, we were not able to generate state-of-the-art results.

Acknowledgments

This work has been supported by the Volkswagen Foundation as part of the Lichtenberg-Professorship Program under grant No. I/82806. The project has been supported by the Klaus Tschira Foundation under project No. 00.133.2008. This research was partially funded by the Ministry of Economy under grant TIN2009-14715-C04-01 (KNOW2 project).

References

Eneko Agirre and Philip Edmonds. 2006. *Word Sense Disambiguation: Algorithms and Applications*.

E Agirre and A Soroa. 2009. Personalizing PageRank for Word Sense Disambiguation. In *Proceedings of 14th Conference of the European Chapter of the Association for Computational Linguistics*, Athens, Greece.

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual Web search engine. In *Proceedings of the seventh international conference on World Wide Web 7*, WWW7, pages 107–117, Amsterdam, The Netherlands, The Netherlands. Elsevier Science Publishers B. V.

A.X. Chang, V.I. Spitkovsky, Eric Yeh, Eneko Agirre, and C. D. Manning. 2010. Stanford-UBC entity linking at TAC-KBP. In *Proceedings of TAC 2010*, volume 758, Gaithersburg, Maryland, USA.

David Ferrucci and Adam Lally. 2004. UIMA: An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment. *Natural Language Engineering*, 10(3-4):327–348.

Iryna Gurevych, Judith Eckle-Kohler, Silvana Hartmann, Michael Matuschek, Christian M. Meyer, and Christian Wirth. 2012. UBY–A Large-Scale Unified Lexical-Semantic Resource Based on LMF. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2012)*, pages 580—-590.

Xianpei Han and Le Sun. 2011. A generative entity-mention model for linking entities with knowledge base. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 945–954.

T H Haveliwala. 2002. Topic-sensitive PageRank. In *Proceedings of the 11th international conference on World Wide Web (WWW'02)*, pages 517–526, New York, NY, USA.

Frederick Jelinek and Robert L. Mercer. 1980. Interpolated estimation of Markov source parameters from sparse data. In *Pattern recognition in practice*, pages 381–397.

M Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. *Proceedings of the 5th annual international conference*, pages 24–26.

David Milne and Ian H. Witten. 2008. Learning to link with wikipedia. In *Proceeding of the 17th ACM conference on Information and knowledge mining - CIKM '08*, page 509, New York, New York, USA. ACM Press.

Roberto Navigli. 2009. Word Sense Disambiguation: A Survey. *ACM Computing Surveys*, 41(2):1–69.

Philip V Ogren and Steven J Bethard. 2009. Building test suites for UIMA components. In *SETQA-NLP '09: Proceedings of the Workshop on Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, pages 1–4, Morristown, NJ, USA. Association for Computational Linguistics.

G Salton and C Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management*.

VI Spitkovsky and AX Chang. 2012. A cross-lingual dictionary for English Wikipedia concepts. *Eighth International Conference on Language Resources and Evaluation (LREC 2012)*.

T T. Brants and A. Franz. 2006. Web 1T 5-gram corpus version 1.1. Technical report, Google Research.

Ioannis Tsochantaridis. 2006. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6(2):1453–1484.