Entity Linking

David S Batista
dsbatista@inesc-id.pt

Abstract

This report describes a prototype system for performing Entity Linking. Entity Linking associates named-entities present in a text to an entry that represents it on a Knowlegde Base (KB), but it can also be the case that the recognized entity in text is not present in the KB, in this case the system should also identify that it's in the presence of a new entity that is not part of the KB. This is a challeging problem, mainly due to the ambiguity associated to named-entities. The prototype takes two approaches: one based on machine learning and another baseda graph structure. The experiments done, used the Text Analysis Conference (TAC) Entity Linking task datasets from the 2009 edition.

1. Introduction

Entity Linking is the process of associating an entity mentioned in a text to an entry, representing that entity, in a Knowledge Base (KB). It can be used to automatically augment text with links, and greatly improve a web user experience. For instance, when reading on-line news, if the main actors in the story are linked to a KB, the user can simply mouse-over and learn more about each. Having the collection of documents where entities are linked to a KB can also be used to perform efficient information extraction or question answering.

An important challenge in this task is Word Sense Disambiguation (WSD) [10], that is, finding the correct sense of a string representing an entity, which can be ambiguous, two kinds of ambiguity might occur:

- the same entity can be referred to by more than one name string; *"Praça do Comércio"* and *"Terreiro do Paço"* both refer the same square in the centre of Lisbon; *"Manfred Albrecht Freiherr von Richthofen"* also widely known as the *"Red Baron"*; *"Bermuda Triangle"*, also known as the *"Devil's Triangle"*; *"Roger Bacon"* also known as *"Doctor Mirabilis"*;
- the same name string can refer more than one entity; there are 4 entries in Wikipedia that correspond to organizations named *"James Cook"*, also there is a town named *"James Cook"*; there are 16 entries in Wikipedia referring people named *"James Cook"*;

Linking entities to a KB generally involved two phases. First named-entities must be identified, this is called entity recognition, that is, parsing a text and identifiying potential words or phrases that represent an entity. Second, entity sense disambiguation, the correct meaning of the word or phrase must be determined according to the context of the named entity. Having a roboust system that can handle different genre of text, such as news, blogs, forums also requires a vast and diverse KB. Wikipedia is a on-line encyclopedia maitained by contributors from all over the world, it contains only for the English language, at the time of writing of this report, more than 3.8 millions of articles and a wide range of topics. It is an obivously choice as a KB for an Entity Linking system.

This report is organized as follows: in section 2 previous work on Entity Linking with base on Wikipedia as KB is reviewed; section 3 introduces the Entity Linking Task from the Text Analysis Conference (TAC); section 4 describes the main architecture and modules of the prototype, section 5 describes features and algorithms used; section 7 describes the experiences done and finally in section 8 results and main conclusions are presented.

2. Related Work

One of the first works on desambiguationg entities in an open-domain text using Wikipedia is the work of Bunesco et. al [2]. They build a dictionary of entities based on Wikipedia's structure and define a ranking function based on contextual similarity and on the taxonomy of categories from Wikipedia.

Cucerzan et al.[3] performed named entity disambigution based on Wikipedia. The disambiguation process involves the Vector Space Model (VSM), in which a vectorial representation of the document is compared with vectorial representations of the Wikipedia entities. The feature vector includes the entities in the document, their context, and their category tags.

The Wikify![7] system automatically identifies entities and disambiguates to the correct Wikipedia entry. Entity recognition is done using a dictionary approach based on how often a term is used as a Wikipedia link among it's total occurences. The more often a term was selected as an entity among its total number of occurrences, the more likely it is that it will be selected again. For entity sense disambiguation two mehtods are combined. The first method is based on the Lesk algorithm [6], which identifies the meaning of a word in a given context by measuing the contextual overlap between definitions of other

Table 1: Description of the datasets used

Dataset	Total	NIL	PER	ORG	GPE
Train 2009	3904	2229 (57.1%)	627	2710	567
Test 2009	1500	426 (28.4%)	500	500	500

words in context. They use the current paragrapah as a representation of the context, and Wikipedia as the dictionary. The second approach integrates both local and topical features into a Naive Bayes classifier. For each ambiguous entity, a training feature vector for each of its occurrences inside a Wikipedia link is generated. The set of possible senses is given by the different links. The features are the word and its part-of-speech, a neighbouring context of three words and the corresponding parts-of-speech, and also a global context, specific keywords determined as a list of at most five words occurring at least three times in the contexts defining a certain entity sense.

Milne et. al[9] took another approach by balancing the commonness (prior probability) of a sense with its relatedness to the sourrounding text. Commoness is simply the number of times a word is used for a specific link (ex. *tree* is used 93% to woody plant, 3% to the type of graph and 3% to the computer science concept). Choosing the most common sense leads to problems, therefore, context terms are used to disambiguate. Each candidate sense and context term is represented by a single Wikipedia article. The problem is then reduced to select the sense article that has most in common with all of the context articles. This comparasion of articles is done by the Wikipedia Linkbased Measure [8], which measures the semantic similarity between two Wikipedia pages by comparing their incoming and outgoing links. Combining those two features commoness and relatedness achieves good results in disambiguating terms in a text.

3. Text Analysis Conference

The Text Analysis Conference (TAC) is a series of evaluations and workshops organized to encourage research in Natural Language Processing. It runs since 2009 the *Entity Linking* task. In the *Entity Linking* task participants are provided with a set of golden queries, a Knowledge Base (KB) and a collection of documents. A query consits of an entity name string and an associated document where the name occurs. The KB is based on Wikipedia and contains facts extracted from infoboxes, and a shorter of version of the article's text. The KB is a snapshot of the English Wikipedia from October 2008 and it contains around 800.000 entities.

The goal is, for each query, return the identifier of the entity in the KB, or NIL if the entity is not part of the KB. There is a semantic type associated to each query: *person*, *organization* or *geo-political entity*. Table 1 describes the composition of the datasets. The dataset of each year is divided into trainning and testing data so that machine learning approaches can be applyed, where the test set contains the official queries for each edition.

4. Prototype System

This section describes the prototype developed. It's composed of four main modules: query expansion, candidate generation, candidate ranking and NIL detection. It is also describe the preprocessing of the Knowledge Base (KB).

4.1 Pre-Processing

Two indexes over the KB were created. One only indexes the entity's name string using as unit level name n-grams, it was based on the Lucene's[1] spellchecker index. The second index contains all the information in the KB. A third index over the Document Collection was also created to facilitate the retrieval of a document associated to a query.

4.2 Query Expansion

In order to capture the highest possible meanings of a query string, a dictionary of possible alternative senses was built. The dictionary contains, for a given string, alternative senses like nicknames, alias, acronyms and spelling variations. The dictionary is based on a Wikipedia dump from 2009. Four kinds of mappings were done: Disambiguation Pages, Redirect Pages, Anchor Links and Acronyms. Every article that is either a disambiguation page or redirect page was analyzed. A mapping was done between the article's title and all the possible redirect or disambiguation pages. Another mapping was made using the anchor links presented in every article's text. For instance *[[Lewis Carrol|Charles Dodgson]]* links the word *Charles Dogson* to the article's page of *Lewis Carrol*. An abbreviations and acronym extraction algorithm [12] was applyed to all Wikipedia's articles text. The algorithm was originally made to extract abbreviations from biomedical texts, but it can also be applied to genres of text.

4.3 Candidates Generation

For each query the system extracts all the possible mappings from the dictionary of alternative senses. For every alternative sense, retrieved from the mapping dictionaries, a query is issued to the index over the entities title, retrieving the top 5 candidates. The number of candidates per query depends on how many alternative senses are found in the dictionary of mappings.

4.4 Candidates Ranking

The context of the query's entity (a support document: news text, blog, etc.) and context of the candidates (information from Wikipedia's entry) from the Knowledge Base is the key clue to find the correct entity among the candidates. I ran experiences with a learning to rank approach and a graph based approach to find the correct candidate entity. The features used on those algorithms are described with detail on Section 5.

Pairwise Learning to Rank is a learning to rank approach, it can be seen as binary classification. A classifier is trained in such a way that for a given pair of instances, can distinguish which one is better. For a list of instances, it predicts the relevance of each with a scoring function. The goal is to minimize the difference with respect to the trainnning data. I used the SVMRank by Joachims [5].

Graph Structure explores the occurrences of entities from the Knowledge Base in the support document and vice-versa to build a graph of occurrences. Exploring the structure

[1] `http://lucene.apache.org/`

of the only graph can lead to good results in choosing the correct entity for a query or returning NIL [4].

4.5 NIL detection

I tested two approaches to detect NIL queries. The first is based on the graph structure described before. The system simply checks if the top-ranked candidate according to the score given by the candidate ranking module as In-Degree of 0, in that case the answer is NIL, regardless of it's ranking score. This approach was based on the work of Guo et al.[4], where they report that using the In-Degree to detect NILs leads to good results.

The other approach uses a SVM classifer to detect if the top-ranked candidate is a valid one or the answer should be NIL. The features used were all based on the score given by the ranking module:

Score corresponds to the ranking score attributed by the Ranking module to the top-ranked candidate.

Mean Score the mean score taking into consideration the score of all candidates.

Difference to Mean Score the difference between the top-ranked canidate and the mean score of all candidates.

Standard Deviation the standard deviation of score of all candidates.

Dixon's Q Test for Outliers a test used to identify and reject outliers defined as: $Q = \frac{gap}{range}$ where $gap = x_1 - x_2$ and $range = x_1 - x_{last}$.

Grubb's Test for Outliers another statistical test used to identify and reject outliers defined, here defined as: $\frac{x_1 - mean}{standard\ deviation}$

5. Features

This section describes all the features used to rank the candidates, divided in four groups: Name Similarity, Textual Similarity, Topical Similarity and Graph Structure

5.1 Name Similarity

This set of features capture different characteristics between the candidate's name string and the query's name string.

Exact Match true if the query's name string and the candidate's name string are the same, does not distinct uppercase and lowercase variations.

Query Substring of Candidate true if the query's name string is a substring of the candidate's name string.

Candidate Substring of Query true if the candidate's name string is a substring of the query's name string.

Query Starts Candidate Name true if the query's string name starts the candidate's name string.

Query Ends Candidate Name true if the query's string name ends the candidate's name string.

Candidate Name Starts Query true if the candidate's string name starts the query's name string.

Candidate Name Ends Query true if the candidate's string name ends the query's name string.

Query is Acronym of Candidate true if the query's name string is an acronym of the candidate's name string.

Candidate is Acronym of Query true if the candidate's name string is an acronym of the query's name string.

Dice Similarity the Dice's coefficient between the candidate and the entity's string.

Jaccard Similarity the Jaccard similarity defined as the size of the intersection divided by the size of the union of two sets of strings: the candidate and entity.

Jaro Similarity the Jaro similarity between the candidate and the entity's string.

Jaro-Winkler Similarity the Jaro–Winkler similarity between the candidate and the entity's string.

Levenshtein Similarity the Levenshtein distance between the candidate string and the entity string.

5.2 Textual Similarity

This set represents features that describe the textual context associated to both the candidate and the query.

Cosine similarity the cosine similarity between the query's support document and the candidate's description.

Query In WikiText true if the query's string name is part of the candidate's textual description.

Candidate in Support Document true if the candidate's string name is part of the query's support document.

5.3 Topical Similarity

This set of features is based on topics extraction. Latent Dirichlet Allocation (LDA) [1] was used to represent topics associated to each candidate. The topics model was generated with base on the textual description's of the Knowledge Base. Stopwords were removed, all words were converted to lowercase and stemmed using the Porter Stemmer algorithm [11]. Then for a given support document from a query, a probability distribution of topics is calculated based on the generated model. I used an implementation of LDA [2] with the α parameter set to $50/K$, the β paramter set to 0.01 and the number of topics $K = 100$. The topics distribution were used to calculate the following features:

Topic Divergence the Kullback-Leibler divergence between the query and candidate's topics distribution.

Topic Cosine Similarity the cosine similarity between the vectors of topics of the candidate and the query.

Topic Match true if the topic with the highest probability is the same for candidate's and query.

[2] `http://gibbslda.sourceforge.net/`

5.4 Graph Structure

I followed an approach taken by Guo et al.[4] which exploits the graph structure of Wikipedia to perform named entity disambiguation based on two measures, Out-Degree and In-Degree.

Out-Degree the nodes in the graph consist of: named entities present in the support document that also correspond to entities in the KB and the text articles of the candidates. There is a directed edge from an article node to a name node when the name is mentioned in the article.

In-Degree the nodes are the name string of the candidates entities and the text articles of named entities, which are also entities in the KB, present in the support document. There is a directed edge that links an article to a candidate name string when the article of a context named entity contains that name.

In the experiments done the graph was built over the Knowledge Base instead of the whole Wikipedia.

6. Experimental Validation

This sections describes the results obtained with different algorithms. It is also included in this section a description of the performance of the candidates generations module. Results are presented in terms of accuracy, that is the number of correct queries divided by the total number of queries.

6.1 Candidate Generation

An important step is the retrieval of candidates for each query from the Knowledge Base (KB). It must ensured that, for all the queries or the most part, the correct candidate is among the list of retrieved candidates. A trade-off has to be made between the average number of candidates per query, and the number of queries for which the correct candidate is retrieved.

Table 2 shows the amount of queries for which the correct candidate was retrieved (coverage), the amount of queries in which the correct candidate was not present (miss rate), and the average number of candidates per query.

Table 2: Recall performance

Coverage	Miss Rate	Candidates p/ Query
77.84% (836)	22.16% (238)	27

Experiences were done using the top 30 candidates per sense, instead of the top 5, which lower the miss rate below 20%, but the average number of candidates per query increased to around 150. For some queries more than 800 candidates were being generated which increased exponencially the processing time.

6.2 Baseline

The baseline method simply uses the ranking of candiadtes return by Lucene, and selects the candidate ranked at the top. Results are presented in Table 3. Since the system always choose the first candidate, and assuming that at least one candidate is always returned by Lucene it would be expected that the NIL accuracy would be 0% however for 4 queries no candidates were returned.

Table 3: Accuracy using the Baseline method

PER	ORG	GPE	ALL	NIL
34.0%	34.4%	11.0%	26.5%	0.9

6.3 Graph Structure

I ran three experiments based on the graph structure and using the scores from the In-Degree and Out-Degree measures. The first approach returns the candidate with the highest In-Degree or NIL if the In-Degree is 0 for all candidates. The second approach follows the same principle as the first, ranks the candidateas according to their In-Degree scores, but it will always return NIL if the Out-Degree is 0 regardless of the In-Degree score. The third selects the candidate with the highest Out-Degree or NIL if the Out-Degree is 0 for all candidates. Results are shown on Table 4.

Table 4: Accuracy using the Graph-Structure approach

Approach	PER	ORG	GPE	ALL	NIL
In-Degree$_1$	13.6%	11.0%	5.8%	10.1%	1.6%
In-Degree$_2$	36.0%	36.2%	19.6%	30.6%	74.9%
Out-Degree$_3$	62.4%	52.0%	44.6%	53.0%	35.7%

6.4 SVMRank

All the experiences done with the trained SVM classifer to detect a NIL candidate failed, the classifer was reporting too often NIL candidates, and accuracy results were too low. Therefore all the NIL results reported are based on the In-Degree and Out-Degree measure of the graph structure, that is, if the top-ranked candidate has an In-Degree or an Out-Degree of 0, the system reports NIL.

Several experiments were carried to study the contribution of each group of features. Table 5 shows the results when a single group of features is removed compared with an experiment with all the features on. The variations are minimal when the group of Topical Similarities or Textual Similarities features are removed. There is a higher variation when the Name Similarities group of features is off.

Table 5: Accuracy after removing different groups of features

Features	PER	ORG	GPE	ALL	NIL
All	77.6%	68.2%	66.8%	70.9%	50.9%
-Topical Sim.	77.6%	67.4%	65.2%	70.1%	47.4%
-Textual Sim.	77.2%	66.0%	66.2%	69.8%	50.5%
-Name Sim.	75.8%	66.2%	63.8%	68.6%	46.2%

Another experience was done, where each group of features was turned on, in addition to the graph structure features. The results in table 6 show that Name Similarities and Textual Simialarity features contribute the most.

The results of accuracy for the best run (all group features on) are still low comparing with the results for the best results from

Table 6: Accuracy after adding different groups of features

Features	PER	ORG	GPE	ALL	NIL
Graph	71.4%	59.8%	58.6%	63.3%	35.7%
+Topical Sim.	72.2%	62.8%	61.0%	65.3%	40.8%
+Textual Sim.	74.6%	63.6%	60.4%	66.2%	38.5%
+Name Sim.	76.2%	64.8%	63.4%	68.1%	46.7%

the edition of 2009 of TAC Entity Linking. The worst results are for the NIL queries.

Table 7: Best run compared with best system from the 2009 edition

System	PER	ORG	GPE	ALL	NIL
All Features	77.6%	68.2%	66.8%	70.9%	50.9%
Best of 2009	83.0%	81.0%	84.0%	82.0%	86.0%

7. Conclusions and Future Work

Altough the reported results are reasonable the system still fails to reach close to the state-of-art. The candidate generation module has a high miss rate (more than 20%) and it compromises the whole pipeline of the system. Too much candidates are generated, especially for queries that are acronyms, probably due to the large number of acronyms loaded into the dictionary of mappings. One possible solution is to use only acronyms extracted from the Knowledge Base or, if an expansion of the query is found on the suport document support document use the expansion instead of the ones in the dictionary. Another approach is to use portions of the support document in the query to the Knowledge Base.

It was expected that the graph structure features would give better results, special in identifying the NIL candidates. Using the whole Wikpedia will probably give better results instead of using only the Knowledge Base.

I plan in the future to explore other features such as using named-entities and facts from both the support document and the candidate's description and explore other learning to rank algorithms. Also reviewing the features and the process of trainning the SVM classifer to detect a NIL candidate, other systems report good results using such approaches for detecting a NIL query.

8. Acknowledgments

I would like to thank Ivo Anástico for providing the datasets already splitted for trainning and testing and also for the spellchecker index code. I would also like to thank Bruno Martins for provinding the code to carry out experiences with the graph structure approach.

References

[1] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3(4-5):993–1022, 2003.

[2] Razvan Bunescu and Marius Pasca. Using Encyclopedic Knowledge for Named Entity Disambiguation. In *Proceesings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-06)*, pages 9–16, Trento, Italy, 2006.

[3] Silviu Cucerzan. Large-Scale Named Entity Disambiguation Based on Wikipedia Data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 708–716, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[4] Yuhang Guo, Wanxiang Che, Ting Liu, and Sheng Li. A Graph-based Method for Entity Linking. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1010–1018, Chiang Mai, Thailand, November 2011. Asian Federation of Natural Language Processing.

[5] Thorsten Joachims. Optimizing Search Engines using Clickthrough Data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 133–142, New York, NY, USA, 2002. ACM.

[6] Michael Lesk. Automatic Sense Disambiguation Using Machine Readable Dictionaries: How to tell a pine cone from an Ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation*, SIGDOC '86, pages 24–26, New York, NY, USA, 1986. ACM.

[7] Rada Mihalcea and Andras Csomai. Wikify!: Linking Documents to Encyclopedic Knowledge. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, CIKM '07, pages 233–242, New York, NY, USA, 2007. ACM.

[8] David Milne and Ian H. Witten. An Effective, Low-Cost Measure of Semantic Relatedness Obtained from Wikipedia Links. In *In Proceedings of AAAI 2008*, 2008.

[9] David Milne and Ian H. Witten. Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, CIKM '08, pages 509–518, New York, NY, USA, 2008. ACM.

[10] Roberto Navigli. Word Sense Disambiguation: A Survey. *ACM Comput. Surv.*, 41, February 2009.

[11] M. F. Porter. Readings in information retrieval. chapter An Algorithm for Suffix Stripping, pages 313–316. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

[12] Ariel S. Schwartz and Marti A. Hearst. A Simple Algorithm for Identifying Abbreviation Definitions in Biomedical Text. In *Pacific Symposium on Biocomputing*, pages 451–462, 2003.