NUS-I2R: Learning a Combined System for Entity Linking

Wei Zhang[†] **Yan Chuan Sim**[‡] **Jian Su**[‡] **Chew Lim Tan**[†]

[†]School of Computing
National University of Singapore
{z-wei, tancl}
@comp.nus.edu.sg

[‡] Institute for Infocomm Research
{ycsim, sujian}
@i2r.a-star.edu.sg

Abstract

In this paper, we report the joint participation of NUS and I2R team in Knowledge Base Population at Text analysis conference 2010. For Entity Linking, we analyze IR approaches and SVM classification in the disambiguation stage and develop a supervised learner for combining these approaches. The combined system performs better than the individual components and achieves results much better than the median. Furthermore, according to our error analysis, quite some errors are caused due to the different Wikipedia version is used, which hinder our system to show significant better performance.

1 Introduction

The aim of Knowledge Base Population (KBP) track at Text Analysis Conference (TAC) 2010 is to automatically discover information about named entities and to incorporate this information in a structured Knowledge Base (KB). The task has been broken down into two sub tasks: Entity Linking and Slot Filling. We participate in the first sub task.

Given a knowledge base and a document collection, the Entity Linking task is to determine for each name string and the document it appears in, which knowledge base entity is being referred to, or if the entity is not present in the reference KB.

KBP track 2010 is a follow-up to the KBP evaluation at TAC 2009. The Entity Linking sub task has been explored by several researchers. The general approach for Entity Linking consists of two stages: name variation and entity disambiguation. Name variation is used to find variations for each entity in KB and to generate an entity candidate set. Entity disambiguation is to link an entity mention with the real world entity it refers to. The crucial component of Entity Linking is the disambiguation process. Varma et al. (2009) reported a disambiguation algorithm to rank the entity candidates using a search engine. Han and Zhao (2009) ranked the candidates based on BOW and Wikipedia semantic knowledge similarity. Zheng et al. (2010) proposed a learning to rank algorithm for disambiguation. Zhang et al. (2010) used an SVM classifier for Entity Linking. Dredze et al. (2010) used the ranking SVM algorithm as the candidate ranker.

In this paper, we describe NUS and I2R joint participation in TAC 2010 where we used a combined Entity Linking system. Based on the similarity between the contextual information of document and entities in KB, we develop several entity linking systems using different IR approaches (Lucene and Ranking SVM) and SVM classification. Finally, these systems are combined using a supervised learning method.

The remainder of the paper is organized as follows. In section 2 we detail our algorithm including name variation and entity disambiguation. Section 3 describes the experimental setup and results. Finally, Section 4 concludes the paper.

2 Algorithm

In this section we describe our algorithm for both challenges of Entity Linking: name variation and entity disambiguation.

2.1 Name Variation

The aim of Name Variation stage is to build a Name Dictionary that contains the name variations

of entities in KB and to generate a candidate set using this Name Dictionary.

Name Dictionary Creation. We use Wikipedia to build our Name Dictionary since Wikipedia is the largest encyclopedia in the world and surpasses other knowledge bases in its coverage of concepts and up-to-date content. We obtain information from Wikipedia using Java Wikipedia Library [1] (Zesch et al. 2008).

Firstly, we map the entities in KB to the entity page in Wikipedia. Next, we retrieve the corresponding redirect pages, disambiguation pages and Wikipedia pages containing the anchor text. Finally, the titles of the redirect pages and disambiguation, as well as the anchor text are used to construct the Name Dictionary. In Name Dictionary, the KB entities are indexed by the name variation string.

Candidates Generation. Using the Name Dictionary created, we can retrieve the candidate entities in the KB that share the same query mention. Moreover, if the name in the query is an acronym, expanding it can effectively reduce the ambiguition of the mention. Thus, before the retrieval process, we expand the acronym queries from the document where the acronym is located.

As Schwartz and Hearst's (2003) algorithm only allows expansions that are in parenthesis adjacent to the acronym or acronym in parenthesis adjacent to the expansion (e.g. *Israeli Air Force (IAF) or IAF (Israeli Air Force)*), we extend it to a more robust algorithm that can find expansions in the whole document.

2.2 Entity Disambiguation

The disambiguation stage is to link the mention with the KB entity it refers to in the candidate set. If the entity to which the mention refers to is not present in KB, *nil* will be returned.

In this Section, we will describe three systems for disambiguation, as well as a combined system.

2.2.1 Lucene[2] System (LS)

In this system, we treat disambiguation as a ranking problem to select a single correct candidate for a query. As the approach in the paper of Varma et al. (2009), we use Lucene to index the Wikipedia text of the candidate entities. Each candidate entity is indexed as a separate document. In query formulation process, we extract all the paragraphs that contain the query mention, and remove the stop words. After that, we form a Boolean "OR" query of all the tokens. Finally, this query is given to the candidate entity index and the relevance score for each entity is calculated by Lucene. The entity with the highest score is considered as the answer.

2.2.2 Supervised Systems

Ranking SVM[3]. Supervised machine learning methods are also popular for ranking problems. We use a ranking SVM algorithm (Joachims, 2002) for disambiguation. In our learning framework, the instance is formed by a list of feature vectors. Each feature vector depends on both the query and a candidate entity. For training instance, we define an ordered constraint, where the score for the feature vector of correct candidate is greater than the scores for the feature vectors of incorrect candidate. The ranking SVM approach is to learn a ranker where the correct candidate should receive a higher score than all other candidates. During testing step, the score for each entity in candidate set is given by the supervised SVM ranker.

SVM Classification. We can also consider disambiguation as a classification issue: deciding whether a mention refers to an entity using an SVM binary classifier. In this learning framework, the training or testing instance is formed by *(query, entity)* pair. The instance is *positive* if the entity is the correct entity, otherwise it is *negative*. Based on the training instances, a binary classifier is generated using SVM learning algorithm. During disambiguation, *(query, entity)* is presented to the classifier which then returns a class label and the corresponding score.

Each *(query, entity)* pair is represented by a feature vector described below.

Features for Ranking and Classification. We selected features for both ranking SVM and SVM classification which have been shown to be helpful in previous works and tasks.

> Exact Equal Surface. The feature is 1 if the mention in query is same as the title of the candidate. Otherwise, the feature value is set to 0.

[1] http://www.ukp.tu-darmstadt.de/software/jwpl/

[2] http://lucene.apache.org/java/docs/

[3] http://www.cs.cornell.edu/People/tj/svm_light/svm_rank.html

Start With Query. The feature value is 1 if the title string of the candidate starts with the mention string of the query and the Exact Equal Surface feature is 0. Otherwise, the feature value is set to 0.

End With Query. The feature value is 1 if the title string of the candidate ends with the mention string of the query and the Exact Equal Surface feature is 0. In other case, the feature value is set to 0.

Equal Word Num. The feature value is the number of same words between the title string of candidate entity and the mention string of the query.

Miss Word Num. The feature value is the number of different words between the title string of candidate entity and the mention string of the query.

Bag of Words (BOW). We use token-based features to measure the similarity between the query document and the Wikipedia text of candidate entity. The cosine similarity metric (standard tf.idf weighting) is used.

Similarity Rank. The feature value is the inverted rank of candidate's tf.idf weight in the candidate set.

All Words in Text. The feature value is 1 if all words in the title of candidate exist in query document. Otherwise, the feature value is 0.

Word Category Pair. We consider word-category pairs as a feature class, i.e., all *(w,c)* where *w* is a word from Bag of Words of document and *c* is Wikipedia a category to which candidate entity belongs.

NE Number Match. The feature value is the number of the same named entities appearing in the query document and the Wikipedia text of candidate.

NE Type. This feature is to guarantee that the type of entity in document (i.e. Person, Geo-Political Entity and Organization) is consistent with the type of entity in KB.

Country in Text Match. The feature value is the number of same countries appearing in the query document and Wikipedia text of candidate entity.

Country in Text Miss. The feature value is the number of countries that appear in the query document but do not appear in the Wikipedia text of the candidate entity.

Country in Title Match. The feature value is the number of same countries appearing in the title of candidate and in the query document.

Country in Title Miss. The feature value is the number of countries that appear in the title of candidate but do not appear in the query document.

City in Title Match. The feature value is the number of same cities appearing in the title of candidate and in the query document.

We use case sensitive string comparisons for the features.

Based on the ranking SVM and SVM classification models using the above features, we develop two supervised systems for disambiguation.

Ranking First System (RFS). In this system, we rank the candidates using the ranking SVM model, and the entity with highest rank is chosen as the answer. As this model always chooses the highest ranked entity as the answer, it does not return *nil* unless the candidate set is empty. Thus, we use SVM classification model to validate whether the highest ranked candidate is the true target entity. The pair of query and highest ranked candidate is given to the binary classifier. If the class label is *positive*, then we return the entity as the answer. Otherwise, *nil* will be returned.

Classification First System (CFS). In this system, we treat the ranking problem as classification. We use the SVM classification model to decide if each *(query, entity)* pair is *positive*. There may be more than 1 candidate that is labeled *positive*. Therefore, we employ the ranking SVM model to rank the *positive* candidates and the entity with the highest rank will be chosen.

2.2.3 Supervised combination

After developing the three systems (LS, RFS and CFS), we combined them into a final system – combined system (CS) using a supervised method. In this method, a three-class classifier is used to judge which systems (LS, RFS or CFS) should be trusted. SVM is chosen since it is state-of-the-art

Runs	All Queries	Non-Nil	Nil	ORG	GPE	PER
CS	**0.7938**	0.6353	**0.9252**	0.7960	0.6876	0.8975
RFS	0.7929	0.6333	**0.9252**	0.7960	0.6849	0.8975
CFS	0.7907	**0.6716**	0.8894	0.7947	0.6796	0.8975
Median	0.6836	-	-	0.6767	0.5975	0.8449
Highest	0.8680	-	-	0.8520	0.7957	0.9601

Table 1: Micro-Averaged Accuracy of Our Runs, Median and Highest

machine learning algorithm. The three features used in this module are the scores given by the three systems for their answer. The classes are the three systems.

3 Experiments and Discussions

3.1 Experimental Setup

Prior to the experiment, we perform pre-processing on the data. In particular, we perform Named Entity Recognition using an SVM based system trained and tested on ACE 2005 with 92.5(P) 84.3(R) 88.2(F). In addition, we use an SVM based coreference resolver trained and tested on ACE 2005 with 79.5%(P), 66.7%(R) and 72.5%(F).

For our implementation, we use SVM^{Light} [4] learner developed by Joachims (1999). The model is trained with default learning parameters. The features' values are normalized to [0,1] to avoid noise caused by extreme values.

Corpora. The training data of KBP 2010 for Entity Linking has 3,904 newswire queries and 1,500 web queries. We select 1,500 of them as our training data for supervised combination and the remaining queries are used for training both ranking SVM and SVM classification. Also, we use the data automatically created by the approach of Zhang et al. (2010) for training the SVM models. The testing data for Entity Linking this year contains 2,250 queries.

Wikipedia data[5] can be freely obtained for research purposes. It is available in the form of database dumps that are released periodically. We use Wikipedia data to augment the given KB data. This allows us to derive name variations mentioned in Section 2.1, and then to build our Name Dictionary. Furthermore, the Word Category Pair feature is based on Wikipedia's category information. The version we used for our experiments was released on Sep. 02, 2009.

Evaluation. The measure used in KBP-10 to evaluate the performance of entity linking is micro-averaged accuracy: the number of correct links divided by the total number of queries.

3.2 Submissions and Results

We submit three runs with different approaches. The description of our runs is as below:

- *Run 1:* Combined System. (CS)
- *Run 2:* Ranking First System (RFS)
- *Run 3:* Classification First System (CFS)

Sixteen teams submitted a total of 46 runs to the TAC 2010 Entity Linking task. Table 1 shows us the results of our runs, as well as the highest and median of participants. The micro-averaged accuracies are provided for *Non-Nil* and *Nil* queries and each entity type. We obtain the following conclusions from the results:

1) The combined system outperforms the individual components and is much better than the median of participants.
2) In the KBP data set, the ambiguations of different entity types are different, which affects the system's performance. *GPE* is most ambiguous and *PER* is least ambiguous.

Although our system achieves results better than the median, its performance is 7.42% lower than the best system. Our system performs at 85.1% on the 2009 KBP testing data, but its performance for the testing data this year is reduced. According to our error analysis, quite some errors are caused due to the different Wikipedia version is used. The KB in Entity Linking is derived from Wikipedia released in Oct. 2008. As mentioned in Section 2.1, to build the Name Dictionary, we map the entities in KB to the entity pages in Wikipedia released in

[4] http://www.cs.cornell.edu/People/tj/svm_light/
[5] http://download.wikipedia.org

Sep. 2009. Different versions of Wikipedia cause much mapping failure.

4 Conclusion

This paper describes our NUS-I2R system in detail for TAC 2010 Entity Linking task. We explored two IR approaches (Lucene and Ranking SVM) and SVM classification model for Entity Linking. Meanwhile, a large feature set which can represent a wide range of information is defined for supervised learning. We propose a supervised learning method to combine the different systems. The combined system outperforms the individual components.

References

M. Dredze et al. 2010. Entity Disambiguation for Knowledge Base Population. In *Proceeding of 23rd International Conference on Computational Linguistics (COLING 2010), Beijing.*

X. Han and J. Zhao. NLPR_KBP in TAC 2009 KBP Track: A Two-Stage Method to Entity Linking. In *Proceedings of Test Analysis Conference 2009 (TAC 09)*

T. Joachims. 1999. Making large-scale svm learning practical. In *Advances in Kernel Methods - Support Vector Learning*. MIT Press.

T. Joachims. 2002. Optimizing search engines using clickthrough data. In *Knowledge Discovery and Data Mining (KDD).*

A. Schwartz and M. Hearst. 2003. A Simple Algorithm for Identifying Abbreviation Definitions in Biomedical Text. In *Pacific Symposium on Biocomputing*, Volume 8, pages 451-462

V. Varma et al. 2009. IIIT Hyderabad at TAC 2009. In *Proceeding of Test Analysis Conference 2009 (TAC 09).*

T. Zesch, C. Muller and I. Gurevych. 2008. Extractiong Lexical Semantic Knowledge from Wikipedia and Wiktionary. In *Proceeding of the Conference on Language Resources and Evaluation (LREC)*, 2008.

W. Zhang, J. Su, C. L. Tan and W. Wang. 2010. Entity Linking Leveraging Automatically Generated Annotation. In *Proceeding of 23rd International Conference on Computational Linguistics (COLING 2010)*, Beijing.

Z. Zheng, et al. 2010. Learning to Link Entities with Knowledge Base. In *Proceeding of NAACL 2010.* Los Angeles, CA