THUNLP at TAC KBP 2011 in Entity Linking

Yu Zhao, Weipeng He, Zhiyuan Liu, Maosong Sun
State Key Laboratory of Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology
Tsinghua University, Beijing 100084, China
{zhaoyu.thu,heweipeng,lzy.thu}@gmail.com, sms@tsinghua.edu.cn

Abstract

Entity Linking is to link a name string from plain-text documents to the corresponding entry in given knowledge base. In this paper we demonstrate our entity linking system for TAC KBP 2011 Track. Our system implements pairwise and listwise learning to rank methods to create a ranking list of candidates with several kinds of features, including context similarity, term frequency, key entity extraction and WikiPage information. We participate in entity linking and cross-lingual entity linking task. We use random forest to validate the top 1 candidate recommended by our system. We achieve a performance of 72.9% F1 measure for both two tasks.

1 Introduction

Entity linking is to link a phrase from plain text documents to an entry in the given Knowledge Base(KB). When processing a document, it is of great importance to extract the named entities from content text because these entities often contain key information of the document. However, some entities are represented by ambiguous words. For instance, the word “Washington” may refer to distinct entities such as the American president *George Washington* or *Washington State* or otherwise *Washington D.C.*. Entity linking solves the disambiguation problems and refer the query string to a determinate KB entry.

In Knowledge Base Population Track(KBP) at TAC 2011, there are two different entity linking tasks namely Mono-lingual Entity Linking and Cross-lingual Entity Linking. Both of the tasks require a system to automatically assign an ambiguous query word to an entity in the knowledge base, or to NIL if none of the entity in the knowledge base can be associated with the query. The knowledge base provided by KBP is a set of articles from an October 2008 dump of English Wikipedia, where each article describes an entity using both unstructured text and structured infoboxes. The query words and context of Mono-lingual Entity Linking task are English, while those of Cross-lingual Entity Linking task are partial English and partial Chinese.

In KBP 2011 there are two main challenges different from those tasks in previous years. The first challenge is that it is required to cluster all NIL entities. Entities not existing in the knowledge base are no longer considered to be the same. The second challenge is that the new cross-lingual entity linking task uses bilingual corpus and multi-language queries, but only English database is provided.

This paper is organized as follows. We first introduce the framework of our entity linking system in Section 2. Then we describe the system modules and our technical approach in detail in Section 3. In Section 4 we discuss the the experimental results. Finally, we conclude in Section 5.

2 Overview

In this section, we present the architecture of our entity linking system. As shown in Figure 1, our submitted system consists of four component modules, including candidate generator, feature extractor, learning to rank module and target entity validation module.

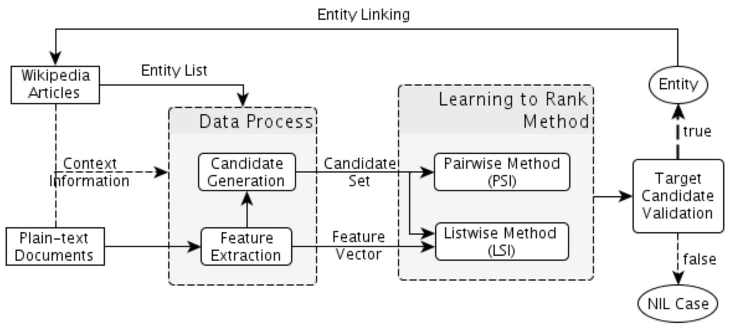

Figure 1: System framework for entity linking

The first two modules function as the data processor. They collect a large number of Knowledge Base entries into a SQL database. Both of these data processing modules take a name string and its background document as query input. For each query, a group of KB entries is drafted by candidate generator. Each query-candidate pair is assigned with a numerical vector by the feature extractor to represent its internal semantic relation. Then we implement learning to rank method to create a ranking list of candidate entries. The entry with the highest ranking score is recommended as the target entity. In consideration of NIL cases, we utilize a validation module to judge whether the top ranked entry really matches the given query.

With the four modules above, our system will return a target entity or a NIL label when receiving the input query. We will introduce these modules in detail in the next section.

3 The Approach

In this section, we present the strategy of our system in detail. As described above, the whole procedure is divided into four parts. We will introduce them respectively.

3.1 Candidate Selection

As the given KB contains a large amount of possible entries, it is necessary to build a relatively small candidate subset with a limited maximum size. Most of the recent approaches are based on name string matching and consider only about the title of KB entry. We notice that the content of given plain-text documents and KB articles are also useful for discovering potential candidates. Four aspects that are listed below can contribute to our candidate selection procedure.

Name String: As usual, the extent to which the query string matches the KB entry title is considered an important basis for candidate selection. We add entities into candidate list if they meet one of the following conditions:

- Articles that title exactly matches query text.
- Articles that the acronyms of title and query text are the same.
- Articles that the edit distance between title and query text is less than 3.

Background Document: During the preview observation for corpus, we have found that surrounding text of given query contains rich information for entity identification. Occasionally the true target entity just lies in the surrounding sentences. For some acronyms especially, the referred entity may appear in a particular pattern. For instance, the surrounding text of query "NHA" is "National Hockey Association (NHA)". Sometimes a few acronyms did not match the target entity. For example, the acronym of "*Independent Turkey Party*" is "BTP" due to the spelling of Turkish. Without context information, these entities are extremely difficult to find.

Therefore, we extract all the abbreviations from the background document. For each of them, we add the nearest entity around it into the candidate set if the query string is an acronym and exactly matches the abbreviation. Furthermore, entities that have similar name string with the query are included either if they exist in the document headline or main body.

Wikipedia Markups: We can derive some special properties from Wikipedia pages. Firstly, we add articles that are listed on certain disambiguation page which contains the given query text in title. It is a recursion process that deals with hierarchy of ambiguous explanations. For example, the disambiguation page "*John*" contains a hyperlink to "*John (surname)*", which is also a disambiguation page and contains the hyperlink to target entity "*Fritz John*". Then we gather a set of hyperlinks from the previously established database. If an anchor text contains query phrases, we involve the referred entity into the candidate set.

If the size of candidate set is too small, it is necessary to import a refilling stage to include other varieties of possible entities. In out system we set up a minimum threshold K for candidate set size to trigger the refilling stage. Articles that title has only one word different from query text, articles that title acronym starts with query acronym and articles that word rearrangement of title meets an acceptable condition mentioned above are added into the candidate set during this stage.

To prove the effectiveness of our candidate selection module, we have tested our system on KBP 2010 training data. We choose $K = 20$ and achieved 96.1% of total recall. The average length of candidate list for all queries is no more than 26, despite the maximized length reaches one thousand. In the submitted version, we limited the size of candidate set to 500 and the average size drops to 24.

3.2 Feature Extraction

For each query-candidate pair, we compute a normalized vector consisting of about 30 features. They are then sent to the ranking module to optimize the weight of a ranking function. These features can be separated into five categories.

For convenience, we firstly introduce some notations below before describing our feature meanings:

- Q_t is the query string and Q_d is the background document context .
- Q_s is the sentence that contains query string in background document.
- C_t is the title of KB article for candidate c, and C_d is the context.

Context Similarity: Features of this category measure the relevance between Wikipedia page content and the given specific plain-text document.

- **Word Overlap**. This feature calculates the ratio of equivalent terms between Q_t and C_t and the value ranges from 0 to 1. During the feature extraction step, we normalize all the feature values to [0,1] by default.
- **Title Containment**. This feature value equals to 1 if Q_t contains or contained in C_t, otherwise it equals to 0.
- **Title Edit Distance**. This feature calculates the edit distance of Q_t and C_t. The value is normalized by the maximum length of Q_t and C_t, to the range from 0 to 1.
- **TF-IDF Similarity**. This feature calculates the tf-idf similarity of Q_d and C_d by Vector Space Model. We obtain the inverse document frequency by statistical analysis based on the whole KB. In addition, we also take TF only similarity as a feature. They are both effective in predicting the correct entity.
- **Query String Sentence**. This feature calculates the similarity of Q_s and C_d.

Term Frequency: This kind of features concerns about prior probability of different query word sense. Candidates of the more frequent occurrence is preferred to have higher scores.

- **Hyperlink Probability**. To approximate the entity frequency, we count times that a certain term appears as anchor text via the established database. For example, 1528 hyperlinks anchored "Apple" refer to "*Apple Inc.*" while 304 hyperlinks refer to "*Apple*" fruit. The candidate which has more Q_t referenced links tends to have higher feature value.
- **Wiki Default Page**. This feature concerns about whether C_t is the default page of Wikipedia while user input Q_t into the searching bar and returns a boolean value. Without

regarding to statistical regularity, this feature takes more account of prior knowledge of human beings and proves to be very effective. The trained weight of this feature ranks top 5 for all subset of training corpus. And we can get it from database without accessing the web.

Key Entity Extraction: By utilizing a Named Entity Recognizing (NER) tool, our system is capable of analyzing vital entities arise in given document or Wikipedia article. Such key phrases are usually pretty good guides for human annotators. For example, if the source sentence for "Apple" contains entity "*Steve Jobs*", it is very possible that the true target is "*Apple Inc.*". It is reasonable that these features work for machine annotators as well. We also find that some special types of words and expressions can contribute to this procedure.

- **First Wiki Sentence**. This feature values 1 if any entity exists in both Q_d and the first sentence of C_d, otherwise it values 0. Redirected title of hyperlinks in such sentence is also included.
- **Nearest Entity**. This feature values 1 if the nearest entity in the surrounding text of Q_t exists in C_d.
- **Same Entity Amount**. This normalized feature calculates the ratio of the same entity in Q_d and C_d. There is a similar feature that counts for Q_s and C_d.
- **Entity types match**. This feature values 1 if the entity types match for Q_t and C_t.
- **Abbreviations**. This feature values 1 if any form of the abbreviation in Q_s exists in C_t, or vice versa. For example, the context "Delaware County, NY" indicates that "New York""should appear on the target Wikipedia page.
- **Cities and Countries**. These features concern about the co-occurrence of cities and countries, including in Q_t, Q_s, C_t and the first paragraph of Q_d.
- **Comma appearance**. This feature values 1 if C_t contains a comma and the term behind it appears in Q_d. We introduce this feature because such kind of Wikipedia titles usually explain their corresponding entity in detail and fit for disambiguation.
- **Prepositions**. This feature values 1 if the word before Q_t is a preposition and the candidate entity type is GPE.

WikiPage Information: The pages of Wikipedia are well-organized and contain useful information. For entity disambiguation task, Dredze (2010) and Cucerzan (2007) used Wikipedia category information to rank candidates. In our system, we derived some features from Wikipedia Infobox.

- **Infobox Link**. This feature is boolean. Hyperlinks that points back to C_d are taken as proof of relevance.
- **Infobox Fact**. This feature calculates the rate that infobox facts match C_d. The generalized Wikipedia infobox contains important statistics and properties of certain type of entity. We remove those pure numerical facts (e.g., longitude and latitude) to avoid a large denominator.

Cross-Lingual Features: We participate in the Cross-Lingual Entity Linking task as well, which is initiated this year. In this task the queries will include both English queries and Chinese queries, which are extracted from the Chinese Newswire corpus, containing about 1 million documents from Chinese Gigaword. Basically, our strategy for this task is to reuse all features in normal entity linking task by introducing a machine translation module, and append several bilingual features to the feature vector. We have tried two interpreters including Google API and Microsoft translator services. By each of them we calculate a unique feature vector and take the average score as output. We import some attributes of Chinese Wikipedia pages into the database during the pre-processing stage. For example, there are internal hyperlinks on Chinese WikiPage that point to the corresponding English entities. Nonetheless, we did not make use of the context of any Chinese Wikipedia articles. Here we describe some special features for cross-lingual task.

- **Chinese Entity Match**. This feature values 1 if there is a Chinese Wikipedia page that title

Training Data	KBP 2009	KBP 2010
No.1	TF-IDF Similarity	TF-IDF Similarity
No.2	Hyperlink Probability	Query String Sentence
No.3	Default Page	Abbreviations in C_d
No.4	Fact Match	City Match In Q_s
No.5	Title Exact Equality	Country In Title

Table 1: Top 5 significant features for different training data

exactly matches the Chinese query text and Q_t is the corresponding English title.

- **Chinese Default Page**. This boolean feature concerns about whether the Chinese query text is the default page of Wikipedia.

- **Chinese Key Entity**. This function calculate the ratio of recognized Chinese entities in surrounding paragraph of Chinese query appear in English C_d. These entities are translated into English before comparison.

3.3 Ranking

Learning to rank method is one of the most practical ways of information retrieval, because ranking is the fundamental problem of many tasks. There are three basic categories of these methods: pointwise, pairwise and listwise. Each of them has different types of loss function. We implement two kinds of ranking algorithm in our system, one is pairwise and the other one is listwise.

Polynomial Semantic Indexing: The Polynomial Semantic Indexing is a supervised model proposed by Bai et al. (2009). PSI tries to train an optimize low-rank matrix by pairwise learning method, to represent latent concepts of a document and minimizes the loss. It can reduce the complexity of high dimensional feature space.

Given the representation vector of Q_t and C_d, the aim of this model is to learn a function that measuring the similarity. We use TF-IDF weighting to get a normalized distribution on words. For degree $k = 2$, denote that the vocabulary size is D, the weight matrix $W \in R^{D \times D}$, and q_t, $c_d \in R^D$. Then we get the basic function $f^2(q_t, c_d) = \sum_{i,j \in D} W_{ij} q_t^{(i)} c_d^{(j)}$. Analogously, for $k = 3$ we get $f^3(q_t, c_d) = \sum_{i,j,k \in D} W_{ijk} q_t^{(i)} c_d^{(j)} c_d^{(k)} + f^2(q_t, c_d)$. As we can see, the space complexity of function f^k is $\Theta(D^k)$, which is apparently unacceptable for our task.

Thus a low-rank approximation matrix is proposed to fix this problem. The new weight matrix W' is decomposed by two $N \times D$ matrix U and V, where

$$f^2(q_t, c_d) = \sum_{i \le N} (Uq_t)^{(i)} (Vc_d)^{(i)} + q_t^T c_d. \quad (1)$$

Here the $N \times D$ matrices can be comprehended as topic distribution on words. And the matrices for Q_t and C_d are different. It implies that query document and KB candidates article may have respective style.

For higher degree, the weight matrix W' is decomposed by more matrices, where

$$f^{k+1}(q_t, c_d) = \sum_{i \le N} (Uq_t)^{(i)} \prod_{j \le k} (V_j c_d)^{(i)} + f^k(q_t, c_d). \quad (2)$$

The space complexity of f^k drops to $\Theta(kND)$.

PSI model chooses pairwise learning. For each query Q_t, the training tuple contains a positive entity and a negative entity. We train the weight matrix W to make $f(q_t, c_{d^+}) > f(q_t, c_{d^-}) + \varepsilon$. Using gradient descent method, we update U and V if $f(q_t, c_{d^+}) - f(q_t, c_{d^-}) < \varepsilon$:

$$\begin{aligned} \Delta U &= \lambda V (c_{d^+} - c_{d^-}) q^T \\ \Delta V &= \lambda U q (c_{d^+} - c_{d^-})^T \end{aligned} \quad (3)$$

where λ is the learning rate.

ListNet: Pairwise methods meet trouble when the negative training samples are not explicit. For the word "Apple", there are more than 20 different possible senses. It is not clear which one of them should be generated as the negative candidate. One solution is to take each sense as c_{d^-} separately, which may extend the training set size by 20 times and restrain the processing speed. The same problems occur for typical classification methods such as SVM.

The proportion of positive and negative cases distinctly affects the quality of training.

Listwise methods can avoid this problem by taking a ranking list of candidates as training instance. They calculate the feature vector for all possible entities from candidate set and select the top 1 candidate as the target entity. We choose ListNet method, proposed by Cao et al.(2007). Zheng et al.(2009) implemented ListNet for entity linking task for the first time.

For each query Q_t, we have a generated list of candidates C= $\{c_{d^1}, c_{d^2}, ..., c_{d^m}\}$. Let π denote a ranking list of the candidates. We introduce a naive function $f_w(c) = w^T c$ to assign a score for each feature vector of candidate c. The probability of ranking π based on score s can apparently be defined as:

$$P_s(\pi) = \prod_{i=1}^{m} \frac{s^{(i)}}{\sum_{j=i}^{m} s^{(j)}}. \tag{4}$$

Here $s^{(i)} = exp(f_w(c_{(i)}))$. Denote each training instance is given as (π_c, π_y), where π_c is the recommended ranking list by current function and π_y is the ground truth permutation. The basic idea of this algorithm is that given the ranking function f_w, the top k probability of π_y should be higher than other ranking lists.

ListNet takes the KL Divergence as loss function, which is defined as:

$$L_k(y, f_w) = -\sum_{g \in \mathcal{G}_k} P_y(g) \log P_{f_w}(g). \tag{5}$$

Here $\mathcal{G}_k$ denotes the set of all possible top k ranking lists, $P_y(g)$ denotes the probability of ranking list g based on the ground truth score y and $P_{f_w}(g)$ denotes the probability of g based on current function f_w. In our system the ground truth y assign positive candidate with value 1, and negative candidates with value 0.

Then ListNet employs Gradient Decent to perform the optimization. We concern only about the top 1 target entity, so we simply discuss the situation when $k = 1$. The initialized parameter w is updated by $\bigtriangledown w = -\eta \bigtriangledown L(w)$, where η is the learning rate.

3.4 Validation and Clustering

Our system uses Random Forest, proposed by Breiman(2001), as a classifier to judge whether the top 1 candidate entity exactly refer to the query word. From previous steps, we get the top 1 entity from candidate set. However, although it is the most related entity in the knowledge base, this entity may not be what the query word refers to. Therefore, it is necessary to validate the top 1 candidate. The classifier in our system uses all the previous features, along with the ranking score as an additional feature. There are some key features that can determine whether the candidate should be the linked to the query word or not. We prefer decision tree as a classifier at first. Nevertheless, the data contains noises which a decision tree is sensitive to. So we choose Random Forest as the classifier in this step. Random Forest is constructed based on decision trees but uses randomly selected features from the feature vector which makes it robust to noises.

We compared the performance of Random Forest and SVM on non-NIL queries of KBP 2010 training data by a ten-fold cross validation. We found that Random Forest achieves a recall of 91.2% while SVM archives 88.5%. The result indicates that Random Forest slightly outperforms SVM in the validation module.

In the end, our system is required to cluster all the NIL entities. We utilize an ensemble of clustering and validation modules, since they share most of the features. For entity linking task, the query name string is used for clustering. For cross-lingual entity linking task, we also take the nearest Chinese entity to the query as the clustering label. We employ the hierarchical clustering techniques based on simple similarity measures of features and label matching.

4 Evaluation

In this section we will show the result of our system for KBP 2011 entity linking task. Our system need not access the web during the entire operation.

4.1 Data

We use two parts of KBP query sets after eliminating the NIL queries:(1) KBP 2010 Training List, containing 1074 non-NIL queries, of which 335 are ORG, 335 are PER, 404 are GPE. (2) KBP 2009 Evaluation List, containing 1675 non-NIL queries, of which 1013 are ORG, 255 are PER, 407 are GPE.

We also take part in the cross-lingual entity link-

	KBP 2011			KBP 2010		
	Precision	Recall	F-Measure	Precision	Recall	F-Measure
ALL	0.72	0.73	0.73	0.76	0.78	0.77
PER	0.75	0.77	0.76	0.87	0.90	0.88
ORG	0.74	0.76	0.75	0.71	0.78	0.74
GPE	0.68	0.67	0.68	0.69	0.67	0.68

Table 2: Evaluation results of Mono-lingual entity linking for each entity type

	Overall	In-KB	NIL
ALL	0.729	0.674	0.783
PER	0.657	0.613	0.692
ORG	0.755	0.618	0.849
GPE	0.793	0.768	0.834

Table 3: B-cubed+ F-score of Cross-lingual entity linking

ing evaluation. For this task we include the cross-lingual Training List, containing 1001 non-NIL queries, of which 251 are ORG, 331 are PER, 419 are GPE.

4.2 Results

We use KBP 2010 gold standard evaluation queries for testing and KBP 2011 evaluation queries for submission. Table 2 demonstrates the results for mono-lingual entity linking task for each entity type. Results of person and organization are better than that of geopolitical. Table 3 demonstrates the results for cross-lingual entity linking task. Our performance is better on NIL cases than those in knowledge base, which indicates that Random Forest works effectively in the validation procedure.

In Table 1, we list the top 5 significant features for different training data. The result indicates that classic TF-IDF similarity is the most important feature in our system. We notice that the term frequency and some key entities are also useful for this task. This conclusion is reasonable because human annotators basically rely on these features as well.

As shown in Table 4, our system achieves higher than median accuracy on both normal and cross-lingual task, but lower than the best team. The result indicates that we perform better on the cross-lingual task. There are two possible reasons for this result. One is that we add key entity features into the validation module for NIL clustering. The second one is that we appended a re-ranking module when calculating the relevance score. The weights of some vital features are distinctly raised.

In Section 3.4, we have introduced two types of learning to rank model, PSI and ListNet. Although we implement both of them in our system, we only adopt ListNet for this task eventually. There are two main reasons. Firstly, for each query Q_t, PSI needs to train separate decomposition matrices. Therefore the distribution on words of training data is required to be more concentrated. However, there is a variety of query strings in KBP training set. Secondly, PSI is proposed to perform approximation on word-to-word matrix. It focuses on the latent concept distribution on a dictionary, not a feature vector. The number of features for each candidate is much less than the vocabulary size.

To prove that PSI is capable of working on entity linking, we test it on a subset of the ambiguous words generated by Mihalcea(2007). We select 6 most ambiguous words from this subset. Then we collect annotations for different senses of these words from the Wikipedia links. Table 5 shows the word sense disambiguation results using PSI model. It illustrates that PSI outperforms the baseline algorithms.

5 Conclusion

Our system implements pairwise and listwise learning to rank methods to link entities in the given knowledge base. We extract four kinds of useful features to represent the relevance of queries and candidates. We use Random Forest for NIL case val-

Task	Entity Linking	Cross-Lingual Entity Linking
Total Teams	21	10
Highest F1	0.846	0.788
Median F1	0.716	0.675
Our System	0.729	0.729

Table 4: Evaluation results for system submitted to KBP 2011 tasks

	number of cases	MFS	R.Mihalcea (2007)	PSI
channel	383	51.09%	71.85%	84.37%
party	491	68.06%	75.91%	74.39%
bar	1329	47.38%	83.12%	91.13%
atmosphere	1100	54.33%	71.66%	65.27%
degree	1094	58.77%	83.98%	90.13%
stress	841	53.27%	86.37%	88.60%

Table 5: Word sense disambiguation results for PSI

idation and find it robust to noises. We achieve a performance of 72.9% F1 measure for both normal and cross-lingual entity linking. Results of PER and ORG queries are better than that of GPE.

In the future, some improvements for our system are possible. We can make use of those entities that are not in the given KB. We can add entries for them in KB and consider them as possible candidates. High ranking of these candidates can be seen as an evidence of NIL case. We should also add more NIL features to enhance the validation and clustering module.

References

B. Bai and J. Weston and D. Grangier and R. Collobert and K. Sadamasa and Y. Qi and C. Cortes and M. Mohri 2009. Polynomial semantic indexing. *Advances in Neural Information Processing Systems*.

Breiman, Leo 2001. Random Forests. volumn 45 of *Machine Learning* page 5-32, Springer Netherlands.

Cao, Zhe and Qin, Tao and Liu, Tie-Yan and Tsai, Ming-Feng and Li, Hang 2007. Learning to rank: from pairwise approach to listwise approach. *Proceedings of the 24th international conference on Machine learning (ICML '07).* ACM, New York, NY, USA, 129-136.

Dan Gusfield. 2007. Large-Scale Named Entity Disambiguation Based on Wikipedia Data. *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics

Dredze, Mark and McNamee, Paul and Rao, Delip and Gerber, Adam and Finin, Tim. 2010. Entity Disambiguation for Knowledge Base Population. *Proceedings of the 23rd International Conference on Computational Linguistics*. Association for Computational Linguistics.

Mihalcea, Rada 2007. Using Wikipedia for Automatic Word Sense Disambiguation. *Proceedings of NAACL HLT 2007*.

Zheng, Zhicheng and Li, Fangtao and Huang, Minlie and Zhu, Xiaoyan 2009. Learning to link entities with knowledge base. *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 483–491.