A Description of ZZ_INFO_TECH System at KBP 2013

Jie Zhou **Yaoyi Xi**
Data Processing Department
Zhengzhou Information Science and Technology Institute
Zhengzhou City, in Henan province, China
zhoujie.nlp@gmail.com xiyaoyi.nlp@gmail.com

Abstract

In this paper, we report the participation of ZZ_INFO_TECH team in entity linking task for Knowledge Base Population at Text Analysis Conference 2013. Our team submitted 4 runs for this evaluation task. In our system, we adopts some simple heuristic rules and context similarity to filter irrelevant candidates; extracts multiple features and trains SVM Ranking model to realize candidate ranking; computes five common statistics and trains SVM model to identify NIL query.

1 Introduction

Entity Linking (EL) is the task of resolving Named Entity (NE) mentions in the given documents to its corresponding entries in a structured Knowledge Base (KB). The EL system can either return a matching entry or NIL to indicate there is no matching entry. It enriches unstructured documents with links to people, places and concepts in the world. Entity linking is a fundamental building block that supports a wide variety of information retrieval, document summarization and data mining tasks. In the information retrieval domain, for the results of NE queries, EL can introduce facts about an entity in addition to pages that talk about it (Bunescu and Paşca, 2006).

The major challenge in EL is ambiguity. The entity mentions in the documents may be ambiguous for a wide variety of reasons: multiple entities share the same name; entities are referred to incompletely; entity mentions are pseudonyms or nicknames; and are often abbreviated.

In this paper, we describe our EL system at KBP 2013. This system adopts some simple heuristic rules and context similarity to filter irrelevant candidates; extracts multiple features and trains SVM Ranking model to realize candidate ranking; computes five common statistics and trains SVM model to identify NIL query.

2 Data Preprocessing

The Wikipedia infoboxes and entries are taken from an October 2008 snapshot of Wikipedia (LDC2009E58). The knowledge base contains a set of entities, each with a canonical name and title for the Wikipedia page, an entity type, an automatically parsed version of the data from the infobox in the entity's Wikipedia article, and a stripped version of the text of the Wikipedia article.

In our system, we mapped October 2008 snapshot of Wikipedia (LDC2009E58) to newer English Wikipedia (the version of Wikipedia in December 2012) by matching their titles, because original snapshot does not contain the information such as redirects, bold text of first paragraph, et al. Although some Wikipedia entries had been edited, most of ones can obtain correct mappings. We expand the aliases of entities in the knowledge base by using newer English Wikipedia. Moreover, the following treatments are also useful in the phase of data preprocessing.

Data Formatting: The initial knowledge base and source documents are in XML format with user-defined tags. To improve the efficiency of query, we convert these documents to structured index files or database files. In the process of source document formatting, lots of small files can be merged into big files for better file management

(such as "*LDC2010E12/2010/wb/*"), inconsistent XML formats and incorrect paragraph divisions need to be processed.

Name Formatting: The query names and entity names existed in the knowledge base need to be formatted. Note that, some special characters (such as "*Ã*", "*ş*") may lead to mismatch between query name and entity name, and this mismatch is also caused by the difference between American English and British English (such as "*World Health Organisation*" and "*World Health Organization*"). So some necessary transformations are used to realize name formatting.

3 Entity Linking

In our system, the EL task consists of three phases: candidate generation, candidate ranking and NIL detection. The details are described as follows.

3.1 Candidate Generation

Since the EL task involves the determination of whether the entry in the knowledge base is corresponding entity of a given query, a candidate set of entries is generated to avoid matching with all entries. We search query name in the name list of Wikipedia entries, and add the matching entries as candidates. In our system, query expansion and candidate filtering are adopted to generate high-quality candidate set.

Query Expansion: Usually, we consider the full name can provide more unambiguous reference. But many times query names are given in the forms of the acronyms and incomplete names. Therefore we identify the full names of these query names from source documents, and consider them as expanded names.

Candidate Filtering: By matching query names and their expanded names with entity names and aliases in the knowledge base, we can generate a candidate set for each query. But this set contains many irrelevant candidates sometimes. In our system, we design simple heuristic rules to identify and filter non-entity candidates firstly. For example, the candidates, whose names start with special characters (such as "."), contain the character ":" or time expressions, contain pre-defined common's text considered as non-entity tags (such as "*novel*", "*album*"), will be removed from the candidate set.

In a practical application, entity classification methods of Wikipedia entries can be applied to classify all entries into pre-defined NE or non-entity types. The capitalization feature of aliases in incoming links is also utilized to identify large portions of non-entity articles in English Wikipedia (Nothman et al., 2008). The methods based on heuristic rules usually achieve high precision. Heuristic rules can be established more easily than those of other NLP applications, such as NER.

Moreover, we computed the context similarity between the candidate's article and query's source text. A minimum threshold is set to determine whether each candidate is related to the query. If the value of context similarity is less than the minimum threshold (in our system it is set to 0.01), the candidates will be removed from the candidate set. And only the top 10 candidates sorted by context similarity in descending order are reserved.

3.2 Candidate Ranking

We select a single correct candidate for a query using a supervised machine learning ranker. We represent each query by a D dimensional vector x ($x \in \mathbf{R}^D$). We adopt support vector machine for ranking algorithm to realize candidate ranking, and SVMRank tool[1] is used in our system.

(1) Features for Candidate Ranking

In the section, we will introduce the features used in the ranking module. Each feature is listed in Table 1, and some detailed descriptions are given below. Our system uses Stanford Log-linear Part-Of-Speech Tagger[2] and Stanford NE Recognizer[3] to realize English Part-of-Speech tagging and NER.

[1] http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html
[2] http://nlp.stanford.edu/software/tagger.shtml
[3] http://nlp.stanford.edu/software/CRF-NER.shtml

Table 1 The features that used in the candidate ranking module of our system.

ID	Name	Description
1	Context Similarity	The feature value is the TFIDF similarity between the candidate's article and query's source text. The words of predefined parts of speeches (noun, verb and adjective) are reserved as the features of vector space model.
2	Entity Similarity	The feature value is the TF similarity between NEs in the candidate's article and NEs in query's source text.
3	Query Name in Candidate's Article	The feature value is 1 if query name exists in candidate's article.
4	Candidate Name (Alias) in Query's Source Text	The feature value is 1 if candidate name (alias) exists in query's source text.
5	Comment's Text of Candidate Name in Query's Source Text	The feature value is 1 if comment's text of candidate name exists in query's source text. The comment's text of candidate name is the text in the parenthesized expression or after the comma, which is used to resolve the ambiguity between articles sharing a name.
6	Other Words of Candidate Name in Query's Source Text	The feature value is 1 if other words except ones contained by query name exist in query's source text.
7	Match for Entity Type of Query Name and Candidate Name	The feature value is 1 if the entity type of query name is same as that of candidate name (alias).
8	Name Similarities	The feature values are name similarities computed by five types of edit distant methods (Dice, Levenshtein, JaroWinkler, Jaro and Jaccard).
9	Average Name Similarity	The feature value is the average of names similarities above.
10	Word Similarity in the Names	The feature value is computed by using the method (Sim_{word}) below. This similarity considers the same words and the prefix relation, such as query name "*Ed Zwick*" and candidate name "*Edward Zwick*".
11	Exact Match for Query Name and Candidate Name	The feature value is 1 if query name is same as candidate name (alias) exactly.
12	Candidate Name in Query Name	The feature value is 1 if query name contains candidate name.
13	Query Name in Candidate Name	The feature value is 1 if candidate name contains query name.
14	Query Name Starts with Candidate Name	The feature value is 1 if query name starts with candidate name.
15	Query Name Ends with Candidate Name	The feature value is 1 if query name ends with candidate name.
16	Candidate Name Starts with Query Name	The feature value is 1 if candidate name starts with query name.
17	Candidate Name Ends with Query Name	The feature value is 1 if candidate name ends with query name.
18	Acronym of Candidate Name	The feature value is 1 if query name is the acronym of candidate name.
19	Acronym of Query Name	The feature value is 1 if candidate name is the acronym of query name.
20	Known Entity Type of Candidate	The feature value is 1 if the entity type of candidate tagged in the knowledge base is known (PER, ORG or LOC).
21	Same Acronym Name	The feature value is 1 if candidate name and query name are the same acronym.

Word Similarity in the Names: For each word set W_q and W_c, which is composed of the words (except stop words) existed in query name and candidate name, the intersection of two word sets is denoted as $Q = W_q \cap W_c$, and complementary sets are denoted as $L_q = W_q \setminus Q$, $L_c = W_c \setminus Q$.

We define suffix-word sets S_q and S_c. For each word $w_i \in L_q$, if there is a word $w_j^{'} \in L_c$ satisfying the criterion that w_i is the prefix of $w_j^{'}$, the word w_i is considered to belong to the set S_q, namely $w_i \in S_q$. We can also get the set S_c using the same way. Then the word similarity in the names is computed by using follow method.

$$Sim_{word} = \frac{1}{2}(\frac{|Q| + |S_q|/2}{|W_q|} + \frac{|Q| + |S_c|/2}{|W_c|})$$

where $|\cdot|$ denotes the size of set.

(2) SVM Ranking

The correct candidate y should receive a higher score than all other possible candidates $\hat{y} \in Y, \hat{y} \neq y$ plus some margin γ. This learning constraint is equivalent to the SVM ranking algorithm of Joachims (2002), where we define an ordered pair constraint for each of the incorrect candidates $\hat{y}$ and the correct candidate y. Training sets parameters such that $\text{score}(y) \geq \text{score}(\hat{y}) + \gamma$. We used the SVMRank tool to solve this optimization problem.

3.3 NIL Detection

In the process of NIL detection, we build the feature vector based on five common statistic values that are used to determine whether the maximum prediction is suspect, and train SVM model to identify NIL query.

(1) Features for NIL Detection

In this section, five common statistic values are computed. For all candidates $C = \{c_1, c_2, \text{L}\ , c_{|q|}\}$ of query q, the prediction of candidate c_i computed by SVM ranking algorithm is denoted as $score(c_i)$. Then we sort all predictions in descending order, and denote the variable as $X = \{x_1, x_2, \text{L}\ , x_{|q|}\}$ ($x_1 > x_2 > \text{L} > x_{|q|}$). A five-dimension feature vector $F = (f_1, f_2, \text{L}\ , f_5)$ is built to train SVM model and predict whether the query is NIL.

If $|q| > 1$, each feature is computed as following:

- Mean $f_1 = \frac{1}{|q|}\sum_{i=1}^{|q|} x_i$;
- Standard Deviation $f_2 = \sqrt{\frac{1}{|q|}\sum_{i=1}^{|q|}(x_i - f_1)^2}$;
- Mean Difference $f_3 = |x_1 - f_1|$;
- Dixon Testing $f_4 = \frac{x_1 - x_2}{x_1 - x_{|q|}}$;
- Grubbs Testing $f_5 = \frac{x_1 - f_1}{f_2}$;

Dixon and Grubbs testing are used to determine whether the maximum prediction x_1 is suspect.

(2) SVM Classification

In our system, we realized Support Vector Machine (SVM) algorithm by using the toolkit libSVM[4] with linear kernels. The training data is selected from KBP evaluation corpus to train SVM model. The type (in-KB or NIL) is considered as the class label of each instance in the training set. Then trained SVM model is used to predict the target class label (in-KB or NIL) of the test data given only the feature vector.

Given a training set of instance-label pairs $(\mathbf{x}_i, y_i), i = 1, \text{L}\ , l$ where $\mathbf{x}_i \in R^n$ and $\mathbf{y} \in \{1, -1\}^l$, the SVM requires the solution of the following optimization problem:

$$\min_{\mathbf{w}, b, \xi} \ \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i$$

$$\text{subject to} \ \ y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

Here training vectors $\mathbf{x}_i$ are mapped into a higher (maybe infinite) dimensional space by the function ϕ. SVM finds a linear separating hyperplane with the maximal margin in this higher dimensional space. $C > 0$ is the penalty parameter of the error term. Furthermore, $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$ is called the kernel function. There are the following four basic kernels:

[4] http://www.csie.ntu.edu.tw/~cjlin/libsvm/index.html

- linear: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$;
- polynomial: $K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i^T \mathbf{x}_j + r)^d, \gamma > 0$;
- radial basis function (RBF): $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \| \mathbf{x}_i - \mathbf{x}_j \|^2), \gamma > 0$;
- sigmoid: $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma \mathbf{x}_i^T \mathbf{x}_j + r)$.

Here, γ, r and d are kernel parameters.

4 Submissions and Results

In the evaluation of KBP 2013, we have submitted 4 runs for the entity linking task. These runs have differences on which training data is used and whether uses the feature of NE similarity or not. The detailed differences between each system are shown in Table 2.

Table 3 lists the runs that we submitted to KBP 2013. There are 2% improvements while the feature of NE similarity is used (Run 2 and Run 4). Table 4 gives the B^3+F1 scores for different subsets of the queries. Compared with WB (Web blogs and newsgroups) and DF (Discussion Forum) data, NW (Newswire) data has achieved better performance. And for each NE type, GPE type performs better than PER and ORG types.

Table 2 The differences between each system.

Run ID	Training Data	Feature of NE Similarity
1	KBP 2009-2010 data	NO
2	KBP 2009-2010 data	YES
3	KBP 2009-2012 data	NO
4	KBP 2009-2012 data	YES

Table 3 Evaluation for all queries (TAC 2013 data).

ID	micro-average	B^3 Precision	B^3 Recall	B^3 F1	B^3+Precision	B^3+Recall	B^3+F1
1	0.644	0.907	0.467	0.617	0.594	0.342	0.434
2	0.657	0.906	0.485	0.632	0.606	0.363	0.454
3	0.653	0.905	0.478	0.625	0.603	0.356	0.447
4	0.669	0.903	0.504	0.647	0.619	0.385	0.475

Table 4 B^3+F1 scores for different subsets of the queries (TAC 2013 data).

	All	KB	NIL	NW	WB	DF	PER	ORG	GPE
Num.	2190	1090	1100	1134	343	713	686	701	803
Run 1	0.434	0.407	0.457	0.502	0.402	0.339	0.425	0.437	0.439
Run 2	0.454	0.454	0.441	0.520	0.402	0.373	0.447	0.424	0.484
Run 3	0.447	0.437	0.448	0.517	0.403	0.355	0.434	0.421	0.480
Run 4	0.475	0.497	0.432	0.534	0.416	0.405	0.475	0.420	0.520

5 Conclusion

We have gotten deeper understanding about the technology of EL by participating in the evaluation of this task. Furthermore, we also summarized several important factors, which might be able to improve the EL performance, with the help of systematic evaluation.

(1) Previous works on EL have been focused on candidate recall because if the target entity is absent in the candidate set, no ranking method can return the correct result. However, with large candidate sets, the difficulty of candidate ranking is increased and candidate precision will decrease. The technology, which identifies and filters irrelevant candidates under the requirement of high candidate recall, is also important in EL systems (Guo et al, 2013). We can add more language resources except Wikipedia (such as cross-language dictionary) to construct richer alias lists, and adopt more strict query condition (such as exact match including case).

(2) The popularity of name can provide critical factor for EL decisions. Without any other information, the popularity of name can tell that in the document "*Michael Jordan*" will more likely refer to the basketball player "*Michael Jeffrey Jordan*", rather than the less popular Berkeley

professor "*Michael I. Jordan*". Usually, the more popular entity appears more times than a less popular entity in a large text corpus, i.e., more name mentions refer to this entity. If the name is common and refers to the entity with high popularity, we can determine the reference even if there is not the evidence.

(3) Some words existed in the source text are important evidences for resolving ambiguity. For example, there is a mention "*Li Na*" in the document who is a Chinese professional tennis player. Throughout the whole document we can find only one word "*tennis*" related to the identification, but we can still determine that it refers to tennis player "*Li Na*" rather than singer "*Li Na*". Therefore, the technology of core evidence identification related to query name in the source text is necessary to achieve better EL performance.

In our system, there are two main problems summarized as follows:

Firstly, there are some missed or inaccurate processes while generating Wikipedia name list and expanding query name, such as the process of disambiguation pages. We need to take more experiments following Hachey's work (Hachey et al., 2013);

Secondly, we only use basic content features extracted from query's source text and candidate's article. Many features, such as document topic, the popularity of name, structure information expressed by incoming and outgoing links, are not used in our system.

References

Razvan Bunescu, Marius Paşca. 2006. Using Encyclopedic Knowledge for Named Entity Disambiguation. In: Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-06), pages 9-16.

Yuhang Guo, Bing Qin, Yuqin Li, et al. 2013. Improving Candidate Generation for Entity Linking. In: Proceedings of 18th International Conference on Applications of Natural Language to Information Systems, NLDB 2013, pages 225-236.

Ben Hachey, Will Radford, Joel Nothman, et al. 2013. Evaluating Entity Linking with Wikipedia. Artificial Intelligence, 194: 130-150.

Thorsten Joachims. 2002. Optimizing Search Engines using Clickthrough Data. In: Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD), ACM, pages 133-142.

J. Nothman, J. R. Curran, T. Murphy. 2008. Transforming Wikipedia into Named Entity Training Data. In: Proceedings of the Australian Language Technology Workshop, pages 124-132.