ICL_KBP Approaches to Knowledge Base Population at TAC2010

Yang Song, Zhengyan He, Houfeng Wang
Key Laboratory of Computational Linguistics (Peking University)
Ministry of Education,China
{ysong, hezhengyan, wanghf}@pku.edu.cn

Abstract

This paper reports the ICL_KBP team participated in the TAC2010-Knowledge Base Popolation Track. We submitted results for Entity Linking task and Slot Filling task. For Entity Linking, we implemented a simple unsupervised method to select the candidate entities in the Wikipedia Reference Knowledge Base for the given query document which describes the query name-string. For Slot Filling, we treat it as a binary classification problem for each candidate slot value and use type constraint to filter the results. Experimental results reveals that our method reaches the median values of all participants for both tasks.

1 Introduction

The Knowledge Base Population (KBP) Track at TAC 2010 aims at exploring extraction of information about entities with reference to an external knowledge source, Wikipedia Reference Knowledge Base. It includes two main tasks, Entity Linking and Slot Filling. The Entity Linking task is to determine for each query, which knowledge base entity is being referred to, or if the entity is not present in the reference Knowledge Base. A query will consist of a namestring and a document-id in the test collection. Each name-string will occur in the associated document in the test collection. The purpose of the associated document is to provide context that might be useful for disambiguating the name-string. The Slot Filling task involves learning a pre-defined set of relationships and attributes for target entities based on the documents in the test collection. A query in the Slot Filling task will contain a name-string, docid, entity-type, node-id, an optional list of slots to ignore.

For Entity Linking, we first use lucene to index the Wikipedia Reference Knowledge Base. And then we use this index to retrieve entities in the Knowledge Base for each query name-string. Finally, we adopted a similarity computing mehtod to rank all the candidates and use a threshold to determine whether the top-one is related or non-related.

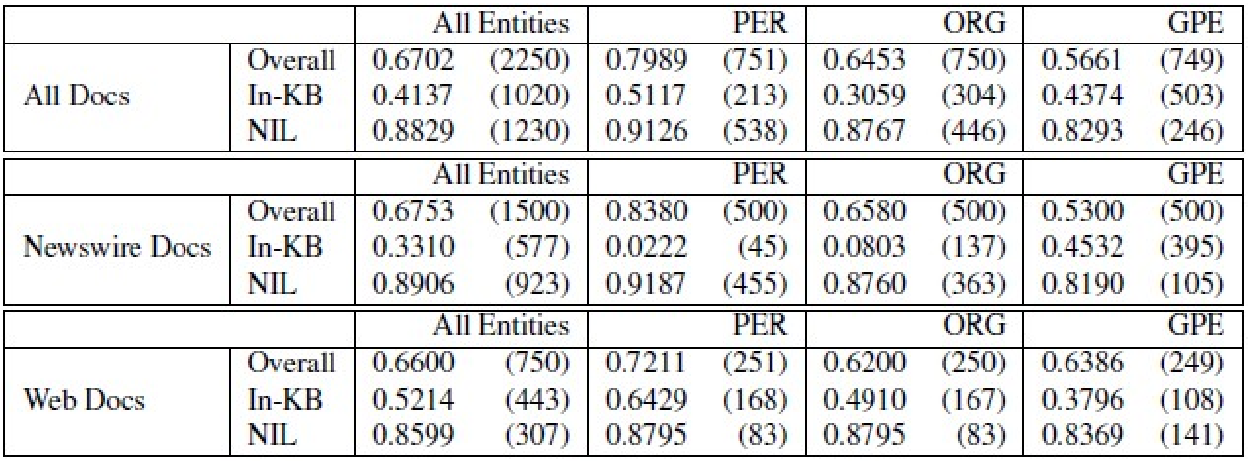
For Slot Filling, first we query the document set with a proximity method. After locating the candidate slot values, we perform binary classification on these values, each classifier coresponding with one slot type. Finally, we pass the results through a type constraint filter to generate the results.

The remainder of this paper is organized as follows. Section 2 introduces our Entity Linking approach. Section 3 gives a detailed description about our Slot Filling system. The conclusions are given in Section 4.

2 Entity Linking Approach

2.1 Preprocessing

Wikipedia Reference Knowledge Base contains 818741 entities (see Table 1), including 116498 GPE entities, 114523 PER entities, 55813 ORG entities and 531907 UKN en-



		All Entities		PER		ORG		GPE	
All Docs	Overall	0.6702	(2250)	0.7989	(751)	0.6453	(750)	0.5661	(749)
	In-KB	0.4137	(1020)	0.5117	(213)	0.3059	(304)	0.4374	(503)
	NIL	0.8829	(1230)	0.9126	(538)	0.8767	(446)	0.8293	(246)
		All Entities		PER		ORG		GPE	
Newswire Docs	Overall	0.6753	(1500)	0.8380	(500)	0.6580	(500)	0.5300	(500)
	In-KB	0.3310	(577)	0.0222	(45)	0.0803	(137)	0.4532	(395)
	NIL	0.8906	(923)	0.9187	(455)	0.8760	(363)	0.8190	(105)
		All Entities		PER		ORG		GPE	
Web Docs	Overall	0.6600	(750)	0.7211	(251)	0.6200	(250)	0.6386	(249)
	In-KB	0.5214	(443)	0.6429	(168)	0.4910	(167)	0.3796	(108)
	NIL	0.8599	(307)	0.8795	(83)	0.8795	(83)	0.8369	(141)

Figure 1: Micro-Averaged Accuracy of our Approach on Entity Linking Task

tities. It's unpractical to make all the entities as candidates for each given query name-string which contains a descriptive document. So we use lucene[1] to index the Wikipedia Reference Knowledge Base. When a query name-string is given, the index is used to select the top 100 entities as candidates. For some query name-strings, we can get the full form of it using wikipedia redirect page. With these full forms about each given query name-string, we can get more accurate results from the knowledge base index.

Entity Type	**# of Entities**
GPE	116498
PER	114523
ORG	55813
UKN	531907
Total	**818741**

Table 1: The count of entities in the Wikipedia Reference Knowledge Base by type assignment.

2.2 Similarity based Candidates Selection

The core task of entity linking is selecting candidate entities. It must be measured by some similarity computing method. Because each entity or query name-string is associated with one document. We use named entities from each document to represent the entities and query name-strings. For doing this, Stanford Named Entity Recognizer[2] is adopted to extract named entities from the Wikipedia Reference Knowledge Base and query documents. We use the number of intersect between candidate entities and the query docuemnts which contains the query name-string as the similarity measure. And we set a threshold empirically to determine whether the top one candidate entity is tagged as the result or NIL. For Entity Linking task (with no wiki text), we only count the named entities which appeared in the attributive slots.

2.3 Experimental Results

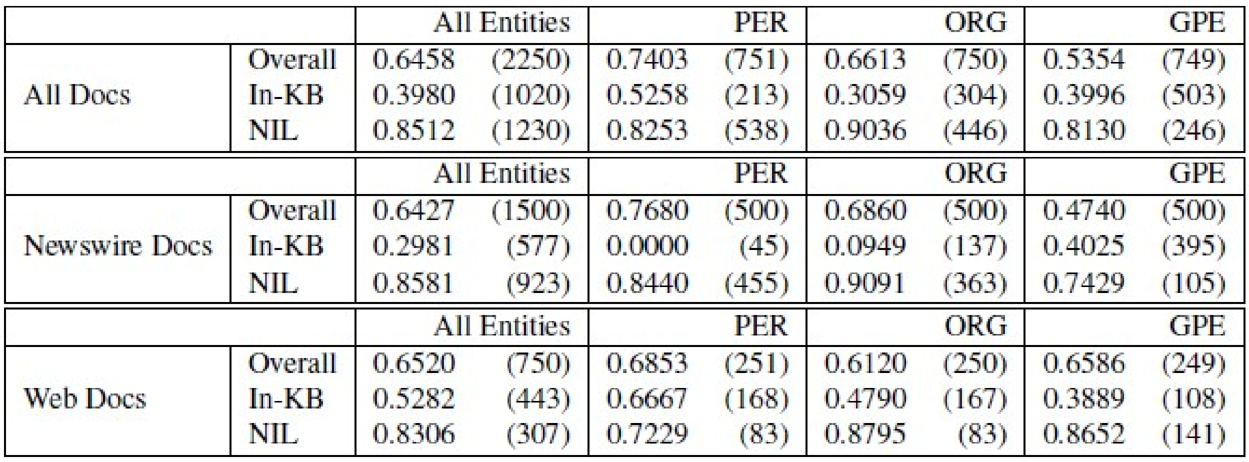
	Highest	Median	Our
2250 queries	0.8680	0.6836	0.6702
750 ORG queries	0.8520	0.6767	0.6453
749 GPE queries	0.7957	0.5975	0.5661
751 PER queries	0.9601	0.8449	0.7989

Table 2: The micro-averaged accuracy comparison between our results and other results on Entity Linking

Experimental results are listed in Figure 1 and Figure 2. The comparison between our results and other results is detailed in Table 2 and Table 3. Our score just reaches the median level of all the participants. There is a large gap between our system and the best one. That's because we only use a simple unsupervised method

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

[2]http://nlp.stanford.edu/software/CRF-NER.shtml



		All Entities		PER		ORG		GPE	
All Docs	Overall	0.6458	(2250)	0.7403	(751)	0.6613	(750)	0.5354	(749)
	In-KB	0.3980	(1020)	0.5258	(213)	0.3059	(304)	0.3996	(503)
	NIL	0.8512	(1230)	0.8253	(538)	0.9036	(446)	0.8130	(246)

		All Entities		PER		ORG		GPE	
Newswire Docs	Overall	0.6427	(1500)	0.7680	(500)	0.6860	(500)	0.4740	(500)
	In-KB	0.2981	(577)	0.0000	(45)	0.0949	(137)	0.4025	(395)
	NIL	0.8581	(923)	0.8440	(455)	0.9091	(363)	0.7429	(105)

		All Entities		PER		ORG		GPE	
Web Docs	Overall	0.6520	(750)	0.6853	(251)	0.6120	(250)	0.6586	(249)
	In-KB	0.5282	(443)	0.6667	(168)	0.4790	(167)	0.3889	(108)
	NIL	0.8306	(307)	0.7229	(83)	0.8795	(83)	0.8652	(141)

Figure 2: Micro-Averaged Accuracy of our Approach on Entity Linking Task, No Wiki Text

which doesn't make the most of training data. At the same time, the experimental results don't drop so much with no wiki text for our simple method.

	Highest	Median	Our
2250 queries	0.7791	0.6347	0.6458
750 ORG queries	0.7333	0.6293	0.6613
749 GPE queries	0.7076	0.4920	0.5354
751 PER queries	0.9001	0.8202	0.7403

Table 3: The micro-averaged accuracy comparison between our results and other results on Entity Linking, No Wiki Text

3 Slot Filling Approach

Slot filling is a standard information extraction task that extracts attributes or properties of named entities. TAC has extended this task to large collections of dataset. It is difficult because we not only have to extract information from free text, but also have to perform disambiguation on entity mentions from different articles.

3.1 Methodology

We interpret this problem as a classification problem. The basic steps of our approach are: during the training process, first select sentences that contains both the query name(or alias) and slot value , then build one classifier for each slot type; during the labeling stage, we first retrieve document path based on some query rules, then we find candidate sentences containing the query word(or alternative name), in the candidate sentences we locate name entity and other formats as candidate slot values, and classify the candidate values for each slot type, finally we post process the classification result through some filters.

3.1.1 Indexing the Document Set

The first step is to index the large document set in order to improve query speed. We index the document set using Lucene package, and build index on the title and text body after removing xml tags.

Furthermore, when we submit queries to the lucene system, we utilize the lucene query syntax of proximity search. It is based on the simple observation that some full person name and organization name do not fully match the name appear in some document. The person name may contains another middle name so the query words will not be adjacent. So when submiting query we set the proximity length equal to the number of query words plus one.

3.1.2 Generating Training Data

During the training stage, we are provided with the query file with query id, query string and its corresponding ground truth, together with the tab format answer file describing the slot type, its value and document path.

First for each answer in tab file, we get the document and find matching sentence containing both the query name (or alternative name) and the slot value of this answer line. For each matching sentence we generate part-of-speech and name entity features according to predefined rules and convert features to libsvm format.

We treat it as over forty binary classification problem, and train one classifier for each slot type.

3.1.3 Matching Alias Names

In order to gain better recall on sentence matching, that is , finding more training instances for SVM, we use some simple rules to deduce the alternative name of the query. If the query name is a full name of a person, we treat sentence that containing one of his first name and last name as matching. If the query name is an organization, the first capital letters of the query name is chained together and serve as its alternative name. This way we can cover more sentences.

3.1.4 Feature Design for Classifier

We tag the matching sentences with Stanford part-of-speech tagger and name entity tagger. We add the relative position such as query first or value first, and whether the two object is adjacent in position. We also add feature that describes the type of the query word and target word, such as one kind of name entity or just noun phrase(for person titles) or just number format. Then we add all words with position and part-of-speech tag with position and name entity tags with position.

3.1.5 Classification and Post Process

During the prediction stage, we are given the query file with its query name and corresponding ground truth. First we retrieve the document with query at a proximity of word count plus one. Then we find matching sentences by considering query name or name alias. In these matching sentences, we generate the part-of-speech tag and name entity tag using Stanford toolkit. This way, we can find person age in all numeric values, person titles in all noun words, entity type in name entities.

Then we feed the matching query name pair and its candidate slot value into each of our classifier to determine the slot type of the candidate slot value. This way we get a list of candidate value for each slot of each query. Finally we post process the result using some rules. These include whether the slot accepts single value or list values, whether the slot value can be a noun, a number, or one of the name entity type. Another issue is some slot value has too many candidate, so we limit the number of the person:spouse and person:title slot.

3.2 Experimental Results

The results are given in Table 4. F-scores of our two submitted result is a little below the median performance.

System	precision	recall	F-score
Top 1	0.668	0.647	0.657
Median	0.214	0.105	0.141
Ours 1	0.111	0.165	0.133
Ours 2	0.132	0.141	0.137

Table 4: The performance of our final system

4 Conclusions

For Entity Linking task, our result only reaches the median values of all the participants. The primary reason is that we only use a simple unsupervised method on this task. Our system doesn't make the most of training data and use very simple document representation and similarity computing methods. Such a system may be considered as a baseline for this task when using unsupervised methods.

For Slot Filling task, our system performs not so good as we cannot handle some of the following problem. We treat it as a binary classification problem, and the positive training instance is sparse and there is a data imbalance problem. The system does not fully utilize the grounded truth to disambigurate the target query if many entities have the same name. The system cannot handle coreference problem, but this is hard for many system as well. We will try to improve our model in future.

5 Acknowledgments

This research is supported by National Natural Science Foundation of Chinese (No.60973053) and Research Fund for the Doctoral Program of Higher Education of China (No.20090001110047).

References

F. Li, Z. Zheng, F. Bu, Y. Tang, X. Zhu, M. Huang. 2009. *THU QUANTA at TAC 2009 KBP and RTE Track.* In Proceedings of Test Analysis Conference 2009 (TAC2009)

X. Han and J. Zhao. 2009. *NLPR_KBP in TAC 2009 KBP Track: A Two-Stage Method to Entity Linking.* In Proceedings of Test Analysis Conference 2009 (TAC2009)

Chieu, Hai Leong and Ng, Hwee Tou and Lee, Yoong Keok. 2003. *Closing the gap: learning-based information extraction rivaling knowledge-engineering methods.* Proceedings of the 41st Annual Meeting on Association for Computational Linguistics (ACL 03)

Soderland, Stephen. 1999. *Learning Information Extraction Rules for Semi-Structured and Free Text* Machine Learning 34,233-272.

Hai Leong Chieu. 2002. *A maximum entropy approach to information extraction from semi-structured and free text* In Proceedings of the Eighteenth National Conference on Artificial Intelligence.