ECNU: Brief System Description of Submission to Knowledge Base Population at TAC 2011

Tiantian Zhu **Wenchao Zhang** **Yue Lu**
Computer Science and Technology Department
East China Normal University
tiantianzhu7@gmail.com

Abstract

This paper briefly reports our submissions to the three tasks in TAC KBP 2011, i.e., Slot Filling (SF for short), Entity Linking (EL for short) and Cross-lingual Entity Linking (CEL for short).

1 Introduction

We participated three tasks in Knowledge Base Population (KBP) Track in Text Analysis Conference (TAC) 2011, i.e., Slot Filling (SF for short), Entity Linking (EL for short) and Cross-lingual Entity Linking (CEL).

There are two subtasks in Slot Filling task this year, i.e., Regular Slot Filling and Temporal Slot Filling. We only focus on the Regular Slot Filling Task in this track. We cast the SF task as a Question Answering task. That is, for each slot given an target entity, our SF system first transforms it into a query and then retrieved passages containing the query like a common QA system. Finally, we adopt various techniques to select appropriate slot value for different slots. Generally, our SF pipeline consists of three main modules, i.e., Query Processing, Passage Retrieval and Slot-Value Selection. This idea comes from the previous work, for example, Lorna and John, 2010, LCC's SF approach (John et al., 2010) and (Daniel et al., 2010). The type of entities provided by organizer can be a person (PER) or an organization (ORG). The KBP 2011 defined 16 slots (i.e., attributes) for organizations and 26 slots for persons. A slot is either single valued (e.g., *date of birth* for PER type) or list valued (e.g., *children* for PER type). For single valued slot, system is expected to return only one slot value. While for list valued slot, system is expected to return one or more different slot values. For each given target entity, the participated system is required to return the slot name, slot values about it as it is and a docid which the slot value comes from. That is, system is expected to return NIL if no novel information for a slot value has been extracted for this slot.

The Entity Linking task aims at linking given entities to the entry in knowledge base (KB) or NIL. If the given entity and target entry are in same language, it is a regular EL task. From this year, the organizers also provide a cross-lingual EL task, where the given entity and target entry are in different languages.

The paper is organized as follows. Section 2 presents the components of our system and the results of our submission. Section 3 describes the system framework and the results of our submission. Section 4 gives a brief description of our work on cross-lingual entity linking.

2 Slot Filling

We adopt a Question Answering system architecture to implement our SF system, which is composed of three sections: passage retrieval, answer extraction and answer selection. In this system, an answer corresponds to one query about the slot value for one slot of a given entity. Unlike a traditional QA system, our SF system does not include a question processing section because the input of our system is just the slot name of one entity rather than a real question. The Passage Retrieval section returns the

top N passages refer to the entity. The Answer Extraction section adopts various techniques, e.g., regular expression, to extract the corresponding substring that seems to be the right slot value for the given entity. Finally the Answer Selection section removes the redundant answers and selects one best answer as the value of the slot.

2.1 Passage Retrieval

Many existing QA systems adopt an information retrieval (IR) component, which treats a question in QA as a query in IR system and returns a list of relevant documents or segments of documents. Our Passage Retrieval section follows this way. We use Lucene(http://lucene.apache.org/) to build our SF system. We first create document index, and then identify those documents that are most likely to contain an answer. We collect the top 100 documents. If there are less than 100 documents retrieved by the system, we would take the all retrieved documents.

Being the first step of our SF system, we expect the system to output as more answer-bearing document candidate as it can. That is, a high recall score is preferred than a high precision at present. However, we found that when we only take the query itself as a system input, the Passage Retrieval module gets a very low recall score. This is not that we expected because no further steps can be performed any more. Therefore, we manually generate some aliases in order to get a higher recall. For example, if the query is an ORG type and is not an abbreviation, we generate its abbreviation as a new input. If the query is a PER type and is a full name, we just use the first name and last name as alias.

2.2 Answer Extraction

The Answer Extraction module takes the passages retrieved from previous model, which are most likely to contain an answer to the slot name, as the input, and selects phrases that are likely to be the expected answer. For each different slot type, we adopt different extraction methods. Roughly, these slot names can be grouped into several categories according to their type and their characteristics. Table 1 lists the different types of the slot names and their corresponding extraction method.

Specifically, we extract slot value candidates by using processing pipelines with multiple techniques including traditional NER, regular expression patterns, Gazetteer-based matching, manually-constructed rules. For example, slot names like *date*, *number*, *website* or *age* are first extracted by using regular expression patterns. Then the system checks if the current sentence has some slot value trigger keywords, such as "*born*", "*birthday*", "*birthdate*", "*birth*", etc. If the slot value is a name from limited name list, for example, *country* or *religion*, we generate a name list to match the slot value. We also adopt Stanford NER to extract named entities, e.g, *persons*, *organizations* and *locations*, from the sentences and then we also check if the corresponding slot value trigger keywords is available in the same passage. Additionally, for different slot name, we also adopt different refinement strategies to make filtering and disambiguation for the purpose of improving precision measure. For example, for slot name *org:subsidiaries*, its extraction rules are as follow:

```
<organization>, a subsidiary of
 <target-entity>
<organization>, a subsidiary of
the <target-entity>
<target-entity>'s <organization>
```

Slot Names	Method
per:date of birth per:age org:website org:founded ...	Regular Expression patterns
per:country of birth per:stateorprovince of birth per:city of birth per:title per:religion ...	Gazetteer-based matching
per:schools attended per:member of per:spouse per:alternate names org:subsidiaries ...	NER tools and manual rules

Table 1: Different slot names and their corresponding potential extraction method

If a sentence in the passage satisfies one of the patterns, then we can extract the corresponding organization as the answer for the slot name. However, some sentences may use aliases instead of the target entity itself. So we also make some deformations about target entity, like abbreviation. Then the patterns also change:

```
<organization>, a subsidiary of
 <entity alias>
<organization>, a subsidiary of
the <entity alias>
<entity alias>'s <organization>
```

Regarding the enumerable slot names, i.e., *country of birth*, *city of birth*, *religion*, etc, a gazetteer (i.e., dictionary) is constructed leveraging on public web resources. We can easily collect a list of country, states, provinces and cities from Internet. Then we use a simple dictionary matching algorithm to extract these candidate values.

There are still some slot names that are hard to extract using the three methods above. One way we used is the method of exclusion. For example, regarding the slot name *per: origin*, we found it difficult to create a common regular expression pattern as well as the slot value trigger keywords. So we make such rules: if the sentence contains country names, but there is no trigger word about birth, country of death or country of residence, then we extract the country as *per: origin*.

2.3 Answer Selection

The previous Answer Extraction module have extracted some candidate answers for each slot name. However, there are many wrong answers and redundant answers for each slot name. The Answer Selection module aims at selecting out the best answers for each slot name. For single valued slot type, we rank the candidate answers according to frequency, that means we compute the number of the same candidate answers, the one with the highest appearance is the slot value. For list valued slot type, we remove the duplicate answers, and make sure each slot value of the corresponding slot name is different from others. If no answer is returned for the slot name, then a NIL is returned as the slot value.

2.4 Results on KBP 2011

Based on the above section, we implement three systems to KBP SF task. Table 2 lists the system brief configuration and its final result on test data set.

Table 2: SF system configuration and results on KBP SF task.

Run	Configuration	R(%)	P(%)	F1(%)
1	without aliases on 09 news data	6.35	2.99	4.06
2	with aliases on 09 news data	17.57	5.27	8.11
3	with aliases on 09 news data and 10 data	18.20	5.24	8.14

3 Entity Linking

The Entity Linking task aims at linking entities to the KB entry or NIL by giving the entity mention string and its source document. We adopt three steps, i.e, Sense Generation,Sense Ranking and NIL Processing to build the whole EL system.

3.1 Sense Generation

The Sense Generation module is to identify as many senses of the entity mention strings as it can. We first use the Stanford NER to extract named entities for `PER`, `ORG` and `GPE` type from the source document. If the extracted entities contain the given target entity mention string, we then record the entity and its corresponding category in Stanford NER. Thus it is a expand of the target entity and we take it as candidate sense. We also search the entity mention string in wikipedia to get as many candidate senses as we can and record the suggested spelling names as well as the names on the Disambiguation Pages. All these are treated as candidate senses.

3.2 Sense Ranking

The Sense Ranking module is to rank the candidate senses and finally identify the most likely sense. Since the Knowledge Base is quite huge, it costs a very long time to search the sense directly. In order to reduce the time cost, we first generate an index of each entry name in Knowledge Base. We found that

there are many punctuation marks in entry names, which disturb the following matching procedure. So we remove the comma and its following text, the bracket and the text between its pair, as well as some other punctuation marks. For the candidate senses, we perform the same removing process as well.

After that, we search the candidate senses throughout index of KB and rank the matching KB entry names. Two ranking methods are used. One method is to simply extract all attributes of one KB entity and match them with the source document. We assign different scores for different KB entry matching as follows: (1) if the attribute appears completely in the source document, assign 3 points; (2) if the class type matches, assign 2 points; (3) if the sub-attribute matches, add 1 point. Finally we sum up all the points and divide the sum with the number of attributes plus 2(i.e, KB name and class name). The KB entry which gets the highest score is the best one we expect.

Another method is to compute the similarity between the source document and the matching wikitxt using $tf.idf$. Terms from source document is denoted by $t1, t2, ...tn$, and the matching wikitxt is denoted by $D1, D2, ...Dm$. $TF(ti, Dj)$ represents the frequency the term ti appears in document Dj, and $IDF(ti)$ represents the frequency the term ti appears in all the documents. For example, for document j, its score is:

$$s(j) = \sum TF(t_i, D_j) * log[(n+2)/IDF(t_i)+1)] \quad (1)$$

Then the wikitxt that gets the highest score is the best KB entry we expect.

3.3 NIL Processing

If no result from the index of KB is returned, then the entity is treated as NIL. We use a filter rule to cluster the NIL entities, that is, if the current NIL is identical with one previous NIL in the document, we cluster them in the same class.

3.4 Results on KBP 2011

We submitted three runs for KBP EL task. The first run adopts the first ranking method without web and wikitxt source knowledge. The second run adopts the second ranking method and does not use web knowledge. The third run adopts the second ranking method and use web knowledge. Table 3 lists the three systems and their results on KBP EL task 2011.

4 Cross-lingual Entity Linking

For the Cross-lingual Entity Linking (CEL) task, we use Google Translate for translation for the reason that Google Translate performs well on `PER`, `ORG` and `GPE` with a acceptable accuracy. With respect to Chinese entity, we extract its corresponding source document and translate it into English. We also translate the Chinese entity itself into English entity. After that, we perform the same runs as we do on English entity linking task above.

We also consider using Chinese Segmentation tool to deal with Chinese text directly, but it is a pity that we cannot find a good Chinese Segmentation tool.

We submitted three CEL systems for KBP 2011. The three systems are based on the three EL systems we built in previous section. Table 4 lists the three systems and their results on KBP CEL task 2011.

5 Conclusion

To be honest, our systems for the 3 tasks are very rude. The methods we use are simple, for the reason that it's the first time we participate in such a huge and international competition, actually we are lack of experience and time, and we still need to learn more knowledge. The result is not good, and there is much space to improve our system, like using some ranking algorithms to improve our ranking result in Slot Filling.

Acknowledgments

The authors would like to thank the organizers for their invaluable support making TAC KBP a first-rank and interesting international event.

References

John Lehmann, Sean Monahan, Luke Nezda, Arnold Jung, and Ying Shi. 2010. *LCC Approaches to Knowledge Base Population at TAC 2010*. In TAC(*Text Automatic Content*) 2010 Workshop.

Daniel Chada, Christian Aranha, Carolina Monte. 2010. *An Analysis of the Cortex Method at TAC 2010 KBP*

Table 3: EL system configuration and results on KBP EL task.

Run	Configuration	R(%)	P(%)	F1(%)
1	ranking method 1, w/t web, w/t wikitxt	48.3	48.3	48.3
2	ranking method 2, w/t web	48.3	46.0	47.1
3	ranking method 2, with web	48.6	48.6	48.6

Table 4: CEL system configuration and results on KBP CEL task.

Run	Configuration	R(%)	P(%)	F1(%)
1	ranking method 1, w/t web, w/t wikitxt	44.5	45.9	45.2
2	ranking method 2, w/t web	39.8	40.8	40.3
3	ranking method 2, with web	43.7	38.1	40.7

Slot-Filling. In TAC(*Text Automatic Content*) 2010 Workshop.

Lorna Byrne and John Dunnion. 2010. *UCD IIRG at TAC 2010 KBP Slot Filling Task*. In TAC(*Text Automatic Content*) 2010 Workshop.

Man Lan, Yu Zhe Zhang, Yue Lv, Jian Su, Chew Lim Tan. 2009. *Which Who are They? People Attribute Extraction and Disambiguation in Web Search Results*. WWW2009, April 20-24, 2009, Madird, Spain.