PolyUCOMP in TAC 2011 Entity Linking and Slot-Filling

Xu Jian[1], Zhengzhong Liu[1], Qin Lu[1], [i]Yu-Lan Liu[2], [i] Chenchen Wang[3]

[1]Department of Computing, the Hong Kong Polytechnic University, Hong Kong
[2]National Tsing Hua University, Taiwan
[3]Tongji University, China

{csjxu, hector.liu, csluqin}@polyu.edu.hk

Abstract

The PolyUCOMP team participated in two TAC-KBP2011 tasks: Regular Entity Linking and Regular Slot Filling.

For the entity linking task, a three-step entity linking system is developed. Similar to some systems in KBP 2010, a list of possible candidates are first selected. Then the best candidate is identified to decide whether a link exists. In addition, a document clustering algorithm is used to group NIL queries. This system incorporates Wikipedia linking information and other textual and contextual features. Our system produces a high answer coverage and accurate linking result. However, the NIL detection system brings significant loss in the final F-score.

For the slot filling task, we developed a system which combines query expansion and pattern-based reasoning. In expanding person queries, name variants are produced using different rules. For organization acronym queries, an abbreviation extraction technique is employed. Manually collected triggers are used for extracting other slot values. Our system ranked above median among all the participating systems.

Introduction

The regular entity linking task is to match a mention string to its corresponding Wikipedia entry, which is referred as the Knowledge Base (KB) node in the task. Additional task this year requires participants to cluster mentions which do not appear in the KB.

Most of the reported works conduct candidate generation followed by candidate selection. Some systems used simple query expansion methods for candidate generation(Chen et al., 2010). Most of the systems used combined sources such as bold text in the first paragraph(Radford, Hachey, Nothman, Honnibal, & Curran, 2010; Varma et al., 2010), Wikipedia redirects and disambiguation pages(Fern, Fisteus, S, & Mart, 2010; Lehmann, Monahan, Nezda, Jung, & Shi, 2010; Radford et al., 2010; Varma et al., 2010), anchor text(Fern et al., 2010; Lehmann et al., 2010; Radford et al., 2010), search engines (like Google)(Lehmann et al., 2010; Varma et al., 2010), local fuzzy search(Radford et al., 2010; Varma et al., 2010), and text matching(Lehmann et al., 2010; Mcnamee, 2010; Radford et al., 2010) to generate candidates.

For candidate selection, some systems treated it as an information retrieval task. Varma et al (2010) used a TF_IDF weighting scheme with query expansion to rank the candidates. Fern et al. (2010) applied the PageRank approach to calculate the rank of entities based on the concurrence information of other entities. Chen et al.(2009) applied the VSM model to KB text. Many systems used a supervised learning approach with various features. Chang et al.(2010) incorporated many syntactic and textual features surrounding the anchor string such as part of speech, bigrams, and trigrams. Some systems have utilized rich features including Wikipedia links, similarity between the candidate string and the mention string and etc.(Lehmann et al., 2010; Mcnamee, 2010). These systems have demonstrated outstanding outcomes in terms of accuracy.

Before selecting the highest ranked candidate as the answer, one important step is to identify “NIL” queries where no node in KB actually matches the mention string. Some systems simply return "NIL" when no candidate is found(Chen et al., 2010; Radford et al., 2010). Others trained a binary classifier (Lehmann et al., 2010; Mcnamee, 2010) or employed heuristics (Chang et al., 2010; Fern et al., 2010) to resolve the problem.

When comes to the slot filling task, previous researchers use query expansion and information extraction techniques (Chen et al., 2010; Chrupala et al., 2010; Surdeanu et al., 2010). Chen et al (2010) combined the bottom-up information extraction with the top-down question answer style pipeline. Besides, they used query expansion and cross-slot reasoning techniques to enhance the algorithms. Chrupala et al (2010) developed a system with a two-stage retrieval module, where document retrieval and sentence retrieval are done in the first stage and relation extraction done in the second stage based on distance supervision technique. Castelli et al. (2010) built an inference engine to derive relations between entities. Bad slots were then filtered out using the cross-document entity co-reference approach. Surdeanu et al. (2010) developed a system which is based on the distant supervision technique.

In this paper, our slot filling system incorporates query expansion and slot filling techniques to find slot values in text. It follows a simple architecture. First, we retrieve documents related to the queries and then preprocess the documents including tokenization, sentence detection, and named entity recognition. Second, query expansion is performed using different techniques including abbreviation extraction and rule-based name variation extraction. Third, we extract slot fillers using entity substitution if available. Others are based on the trigger words manually collected from the English Wikipedia. Then, the document relevance score given to validate the effectiveness of the extracted slot fillers.

The rest of the paper is organizes as follows. Section 2 describes the design and performance analysis of the entity linking system. Section 3 describes the design and the performance analysis of the slot filling system. Section 4 is the conclusion.

The Entity Linking System

Our entity linking system also takes the two step approach to first generate the candidates, and then through the second selection step to obtain the result. As the task this year also requires clustering of the results, an additional clustering step is developed to handle "NIL" queries. The following sub-sections explain our approaches in detail.

1.1 Knowledge Base preprocessing and preparation

In order to make use of the rich linking resources in Wikipedia, a Wikipedia dump[1] with only articles is prepared for use. The matching process is divided into two parts. First, the mention string is linked to Wikipedia articles. Another mapping is done from the KB nodes to the Wikipedia articles. In principle, the Wikipedia articles should be a superset of the KB nodes. However, due to the conflict in different Wikipedia versions, there is a small set of articles in KB nodes which cannot find a mapping in the Wikipedia articles. In our system, this portion of the KB nodes is ignored.

The KB nodes are indexed using Lucene[2] with fields including "Title", "Text", "Node Id", "Node Type"(a value in the set {ORG, GPE, PER, UNK}), and "facts"(The Wikipedia fact slots). The mapped Wikipedia articles are processed using the Wikipedia Miner toolkit[3] described in (Milne & Witten, 2008). The indexing process enables extensive search and mining in the following sub-sections.

1.2 Candidate generation

The primary goal in candidate generation is to achieve a high recall so that we can obtain wide candidate coverage. Precision should be considered if possible, but precision is only a secondary consideration in candidate generation. To achieve a high recall, several sources are used together to get the set of candidates. The sources are listed below:

S 1. Surface Form to Entity Mapping(SFEM)

[1] A Wikipedia dump on 2010/Oct/11 is used in our system
[2] http://lucene.apache.org/
[3] http://wikipedia-miner.cms.waikato.ac.nz/

First of all, the system tries to find all possible "*Senses*" using the Wikipedia Miner. Senses are modeled by Wikipedia pages, they are generated through **Surface Form to Entity Mapping** (Cucerzan, 2007). Surface forms are the mentions of an entity, and entity is modeled by the Wikipedia page, which is also called sense in the Wikipedia Miner system. Surface forms can be page titles or references (Wikipedia anchors) in other Wikipedia pages to this entity.

S 2. Proximity SFEM
If no such pages (senses) are found, the system will try to give *suggestions* by finding page titles. These titles are filtered out by computing their edit distances with the mention string to correct some spelling errors.

S 3. Tracing of Actual Pages
In case the returned page are not article pages, such as redirect or disambiguation pages, the system will follow the links to find article pages with actual contents and add them into the candidate set.

S 4. Candidate Augmentation Through Lucene
To ensure reasonable recall, the system will search the mention string in Lucene if the number of candidates in Step 4 is less than 7. By so doing, sufficient number of candidates is produced for selection.

S 5. Candidates from the source documents**:**
Apart from retrieving candidates directly from the Wikipedia articles, used three methods to find out candidates in the source documents similar to the LCC 2010 system(Lehmann et al., 2010).

(1) **Longer Mentions to** identify longer mentions of the entity name in the source document such as:
Query: "Black Panthers"
Sense: "New Black Panthers".

(2) **Soft Mentions** to identify approximate string with the entity name. This type of mention aims to find alias with different punctuation marks or to correct mis-typed query names. Dice coefficient is used to compute the similarity between strings as in
Query: "Carrie Ann Moss"
Sense: "Carrie-Ann Moss".
The threshold for the Dice coefficient test is set to 0.6 based on observation.

(3) **Acronym Expansion** to identify expanded forms of an abbreviation. Because a large portion of the documents are from newswire, abbreviations usually appear in complete form and they are followed within parentheses for the first time. For the string before the parentheses, the system maps the words' initial letter into query name and allows a gap of at most 2 words. For example, "Convocation of Anglicans in North America" can be mapped into "CANA". There are also cases where the abbreviation are not formed by English, in this case, The Stanford NER tagging is used to extract the named entity right before the parentheses. For example, "BA" is the German abbreviation of "Federal Labour Agency".

For the candidates generated above, each candidate will be found in the Wikipedia articles, and is represented uniquely by its mention string, query id, KB node id (sometimes can be NIL) and Wikipedia page id.

1.3 Candidate selection

Candidate generation in the candidate generation module can introduce a lot of noise into the candidate set especially the work in Step 4 and Step 5. The purpose of candidate selection is to filter out these irrelevant candidates. In our case, candidate selection is done through supervised learning using SVM$^{\text{Rank}}$ [4] and SVM$^{\text{light}}$ [5]. Learning is done through the answer sets from KBP 2009 and 2010 using ten features associated with the candidates as listed in Table 1. These features are either translated to binary values or real values before applied to the classifiers. Out of the 9 features, four are textual features, three are contextual features, one is semantic feature, and one is a confidence score.

[4] http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html
[5] http://svmlight.joachims.org/

Feature	Feature	Data Type	Description
DICE_TEST	Textual	B	Whether the mention string and candidate string pass the DICE_TEST
ACRO_TEST	Textual	B	Whether the mention string is an acronym of the candidate string
SUBSTRING_TEST	Textual	B	Whether the mention string and candidate string are substrings to each other
WEAK_ALIAS	Textual	B	If all the above fail, then this is true
COMMONESS	Contextual	R	Probability of the anchor text refers to this page.
RELATEDNESS	Contextual	R	Semantic similarity of the Wikipedia page to the mention context.
LUCENE_SCORE	Contextual	R	For candidates found in Lucene search, the search score is used.
SAME_TYPE	Semantic	B	Whether the candidate shows the same type in KB nodes and DBpedia or NER tagging
NO_OF_SOURCE	Confidence	R	Number of sources that the candidate string is found

Table 1 Features in Entity Linking and NIL Detection

Textual Features

The **DICE_TEST** feature considers the Dice Coefficient of the query name and a candidate mention in different strategies: In the first strategy, the Dice Coefficient is done on the original strings. We also adopt two additional variation forms to compare only a portion of the long string so that the two comparing strings have the same length. The variation forms are the left aligned and the right aligned strategies. The table below illustrates how these three strategies are applied.

Strategy name	Short string	Long string
Both full length strings	Abott Lab	Abbot Laboratory
Left aligned	Abott Lab	Abbot Laboratory
Right aligned	Abott Lab	Abbot Laboratory

If one of the coefficients exceeds the threshold 0.8, the feature is valid and assigned with a Boolean value (true). The **ACRO_TEST** feature only considers all-uppercase query names, namely, the query names in abbreviation form. There are two rules in the acronym test. The first one requires that some of the initial letters of candidates can form the query name in any order such as "CHT" and "Chunghwa Telecom Co.". The second one requires that the letters of the query name must be found in the same order as that in the candidate string.

The third textual feature **SUBSTRING_TEST** tests the substring matches of the mention string in the source document. The last feature, WEAK_ALIAS, indicates the failure of getting any textual feature.
Although the set of rules seems loose, by using these features in combination, the disambiguation power is still considerably high.

Contextual Features

The context around the mention string plays an important role in determining the mapping. We use a method similar to the highest ranked system LCC(Lehmann et al., 2010) in KBP 2010 by modeling the context using Wikipedia articles. The rich information from Wikipedia linking provides more resources than common term matching. The Wikipedia miner(Milne & Witten, 2008) system is used to model the context around the mention string into Wikipedia concepts (modeled by Wikipedia articles). With the use of this tool, we can compare the similarity between the candidate Wikipedia articles with the context using the internal linking similarity. We adopt the commonness and relatedness measurements described in (Milne & Witten, 2008) and (Cucerzan, 2007) as the primary sources for context similarity.
Commonness feature is the probability that this anchor text will link to the candidate page. The **Relatedness** feature is the semantic similarity of two Wikipedia pages, calculated using the number of links between these two pages. To speed up the

process of link mining, only outwards links are considered in our system. To make use of the relatedness score, the context near the candidate string is modeled through the following steps:

1. The learning based link detector in Wikipedia miner is used to detect links in a window size of 20 near the mention string.
2. The links are ranked based on several criteria such as the relatedness and link probability with the surrounding links(Milne & Witten, 2008). If one link is less related to the surrounding links or it is too popular, it will be given a low score. The judgment is based on a pre-trained model shipped with Wikipedia miner. Only links with a score in the top 10 are reserved.
3. The window size is incremented by 20 if less than 10 links are found. The maximum window size is 80 as a termination condition even if not enough context terms are found.

Lucene score used in candidate generation is also used here as a contextual feature.

Semantic Features

The **SAME_TYPE** feature refers to whether the mention string is the same semantic type as that of the KB node. The Stanford NER tagger[6] is used to identify the a query's semantic types. When verifying the 2009 and 2010 results with the published golden standard, the accuracy only reaches 86%. Therefore, DBpedia is included in the system to improve accuracy. DBpedia is consulted first; NER tagging is used when DBpedia's answer is unknown.

Ranking is then conducted using The SVM^{Rank}system. The C parameter for SVM^{Rank} is set to 0.01 multiplied by the number of training examples. The top ranked candidate is the potential linking node for the corresponding query.

NIL Detection

Another important procedure is to detect the NIL queries. In our system, a binary SVM classifier is trained using the same feature set. SVM^{light} is used to train the model. For the submitted run, the cost factor (-j option in SVM^{light}) for positive and negative samples are set as using the system default value of 1.

1.4 NIL query Clustering system

A new task in KBP 2011 requires that all the output be clustered based on the underlying entities. A sub system that clusters NIL queries is developed. The system is also divided into two parts:

P1. NIL query grouping

As there are a large number of NIL queries in the final result, all the NIL queries are first clustered based on similarity to the query string. There are three tests to determine the string similarity, Dice coefficient test, Acronym test and substring test. The threshold values of these tests is the same as that used in the entity linking part. If a string passs the test with any other string, they are treated as similar, the system go through all the NIL queries and group the similar queries together.

P2. Inner group clustering

In each similar group, the queries need to be further clustered. Because of time constraint, only one similarity measure -- relatedness is used. The system models the context near the mention string as Wikipedia anchors, and finds the similarity between contexts. As the contexts have already been modeled in the entity ranking part, the system only needs to get the existing context for comparison, which makes this step very efficient.
For each context anchor in a document D_i, we assign it the similarity score with the most similar anchor in document D_j. The average of all these scores in document D_i is used as the final similarity score between d_i *and* d_j Use A_{im} to denote the *m*th anchor in D_i, A_{jn} to denote the *n*th anchor in D_j. *R(A, B)* refers to the relatedness score between two anchors, the similarity score is then computed as:

$$sim\big((d_i, d_j\big) = \sum_m \max_n (R(A_{im}, A_{jn}))$$

A simple Hierarchical Agglomerative Clustering (HAC) algorithm is used to cluster the documents. The threshold used for the HAC is set as 3.0 for the submitted run.

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

1.5 Performance and evaluation on entity linking system

Due to time constraint, only two runs with slightly different threshold are submitted. Their performance is listed in **Error! Reference source not found.**. Our system gives a good performance in the first step. In the 2011 test data set, the system achieved 85.5% coverage on Non-NIL queries. However, the final evaluation result shows that the performance is lower than average.

In post evaluation experiments, we found one programming error where the binary classifier and the rank classifier are used in the wrong order. The corrected version of is labeled as Run3 and its performance is listed in **Error! Reference source not found.**. Also, when training the binary classifier, we did not address the severe imbalance between the negative examples and the positive examples. With the added cost parameter "-j" in SVM^{light} set to 3 to correct this imbalance, RUN4's performance has further improved.

	Micro Average	B3 Precision	B3 Recall	B3 F1
RUN1	0.673	0.582	0.634	0.607
RUN2	0.672	0.580	0.633	0.605
RUN3	0.694	0.664	0.619	0.641
RUN4	0.740	0.713	0.654	0.683

Table 2: Performance Evaluation of the Entity Linking System

Given that the highest F1 score among all participants is 0.846 and the median F1 score is 0.716, the system performance is close but still below the median. In the analysis, we found that the NIL detection system is still working poorly even with a penalty to address the data imbalance issue. For a trained model without penalty, there are 400 results classified as NIL by mistake, given that the number of queries that has a real linking in KB is only 1124, 35.5% answers are missed in this way. At the same time, the system also left 178 NIL queries undetected. After giving penalty to the model, there are still 283 queries classified as NIL and 192 NIL missed mistakenly. But, the ranking classifier works a little bit better with penalty: among 841 queries with a link that the system did not classified as NIL. 718 answers are correct, and the accuracy is about 85.5%.

We suspect that the selected features used in the system are not appropriate for NIL queries. On the other hand, we also notice that the NIL classification recall on the 2011 data set (about 64%) is significantly lower than the test result on 2010 or 2009 data set (all above 80%). The differences in data set may also result in the poor performance on NIL detection. This may be because the currently used features are not suitable for NIL queries. There is a need to find discriminating features suitable for NIL queries in the future.

The Slot Filling System

Our slot filling system has three modules. The first is for document retrieval and preprocessing. The second is for query expansion and third is for entity substitution.

1.6 Document retrieval and preprocessing

The entire source text is first indexed by the Lucene package. Initially, when expanding a query, we use Lucene to get the top 50 documents that contain the query name. We then employ the Stanford NLP package[7]including the named entity recognizer tool and part of speech tagger to preprocess the documents. We also used the OPENNLP package [8] tokenizer and sentence detector to tokenize the documents and find the sentence boundaries of the documents.

For a particular set of slots, including *org:emeber_of*, *org:members*, *org:parents*, and *org:subsidiaries*, the Stanford dependency parser is used to harvest related slot values. Before extracting the slot values, the named entities are substituted with one capitalized word. By doing so, the named entities with more than one words will be parsed as one single constituent in the dependency tree. This will help to find the correct slot values.

For other types of slots, for example, *per:title*, *per:charges*, *per:origin*, *org:political/religious affliation*, lists of trigger words are used. These trigger words are manually collected from the English Wikipedia. As for the types of *per:date_of_birth, org:founded, org:dissolved*, etc.,

[7] http://nlp.stanford.edu/

[8] http://incubator.apache.org/opennlp/

we simply use pattern based approach to extract their slot values. For location slots such as *per:city_of_birth*, *per:cities_of_residence*, *org:city_of_headquarters*, we use the geo-name lists to identify city, country and state.

1.7 Query expansion

We employed three methods to expand the queries as it is obviously insufficient to retrieve KBP source documents by merely using a query name.

(1) If the query has Wikipedia redirect pages, the titles of the redirected pages are used as the query expansion.

(2) For a person query, we list different variations of the person's name. For example, given the query Abdul Rahim Noor, we will obtain the variants of this names as shown below:

Abdul Rahim
Rahim Noor
Abdul R. Noor
Abdul R Noor
A. Rahim Noor
A. R. Noor
A R Noor
Noor, A R
Noor, A. R.
A. Noor

After generating these variations, the top 50 documents retrieved by the query name will be used to check the occurrence of these variants. Only those that appear in the text will be selected as the variants That is to say, any of the name variants found in the top 50 retrieved documents will be added to the extended query set.

(3) Person names in the background document are extracted to expand person queries.

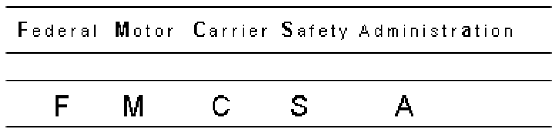
The Stanford named entity recognizer is used to extract all person names in the background document to see if that person name contains the query name. If so, the person name is added to the query extended set; otherwise, we use the Edit Distance to measure the similarity between the person name and query name, and only keep those person names within the threshold.

(4) To find the full names of organization acronyms in the background document given in the query.

To get the full expressions of the organizational queries, the abbreviation extraction technique (Schwartz & Hearst, 2003) is employed. This technique is described as:

(i) long form '(' short form ')'
(ii) short form '(' long form ')'

These <long form, short form> or <short form, long form> pairs are determined by their adjacency to the parentheses. For the sake of slot filling task, only <long form, short form> will be discussed, and the long form and short form are considered adjacent to each other. Using the (i) pattern, candidates for the long form will be recognized. The long form candidates contain contiguous words before the short form. This algorithm starts from the ends of short form and long form and tries to capture the shortest long form that matches the short form. See the figure below:



This algorithm initially starts from the end of the short form and gets the capitalized letter *A* in short form, then it searches *A* in the last word *Administration* of the long form. If *A* is not found in *Administration*, this algorithm will move to the next word *Safety* prior to *Administration*, and check if this word contains the letter. It repeats this process until no matching is found at the beginning of the long form candidates. However, if the letter is found in *Administration*, this algorithm will move to the next letter S and next word *Safety*, and check if *S* is in *Safety*. The whole process stops when it reaches the beginning of the long form and short form. Moreover, this algorithm places a constraint on the <long form, short form> pair that the first character of the word in the long form should be the same as the one of the short form.

By so doing, for the query of TACC, its text is: *The International Atomic Energy Agency's Technical Assistance and Cooperation Committee (TACC) failed to reach a final decision ...*

Using the abbreviation technique, the long form Technical Assistance and Cooperation Committee can be extracted for the acronym

TACC although the word *and* appears in the long form.

1.8 Entity Substitution

For the slots *org:emeber_of*, *org:members*, *org:parents*, and *org:subsidiaries*, the Stanford dependency parser is used to harvest the related slot values. Before extracting the slot values, the named entities are substituted with one capitalized word. Take the *org:parents* for example, the organization query name is replaced by the symbol ORGSUBSID in the sentence. Afterwards, The Stanford named entities recognizer is used to find organization names in this sentence. These organization names serve as the parents of the query candidates. They are substituted with the symbol ORGPARENT. In the next step, the dependency parser is used to obtain relations between ORGSUBSID and ORGPARENT. The dependency relations include:

1.nn: noun compound modifier.

2. appos: appositional modifier.

3. amod: adjectival modifier..

4. prep_of: the preposition word *of*.

5. rcmod: relative cause modifier.

6. poss: possession modifier.

7. conj_and: the conjunction word *and*.

With these dependency relations, triggers words are introduced to build up connection between ORGSUBSID and ORGPARENT. The trigger words involved in *org:parents* contain agency, subsidiaries, subsidiary, affiliation, affiliated company and so on. Using the dependency relations and trigger words, five rules are formulated below:

Rule1: appos(ORGSUBSID, TriggerWord) AND amod(TriggerWord, ORGPARENT)

Rule2: prep_of(TriggerWord, ORGPARENT) AND rcmod (ORGSUBSID, TriggerWord)

Rule3: nn(ORGSUBSID, ORGPARENT) AND nn(ORGSUBSID, TriggerWord)

Rule4: appos(TriggerWord, ORGSUBSID) AND poss(TriggerWord, ORGPARENT)

Rule5: conj_and(ORGPARENT, TriggerWord) AND conj_and (ORGPARENT, ORGSUBSID)

In these rules, organization names are replaced with the tag ORGPARENT or ORGSUBSID and trigger words are replaced with the tag TriggerWord. In each rule, both dependencies must be satisfied will the ORGPARENT be chosen as the value for the slot *org:parents*. Take the following sentence as an example:

An article published Sunday by ***Bernama*** *, the* ***Malaysian government*** *news agency , singles out .. .*

The organization name *Malaysian government* needs to be found as the *Bernama*'s parent organization. Before extracting the *org:parents*, the named entity recognizer is used to obtain the organization names *Bernama* and *Malaysian government*. Then, *Bernama* is replaced with the symbol ORGSUBSID and *Malaysian government* with ORGPARENT. Based on these entity substitutions, the original sentence is changed to:

An article published Sunday by ***ORGSUBSID*** *, the* ***ORGPARENT*** *news agency , singles out ….*

Then the Stanford dependency parser is applied to obtain the relations between the components in the sentence. The dependency relations are listed below:

```
     amod(agency-11, ORGPARENT-9)
    appos(ORGSUBSID-6, agency-11)
```

Similarly, these rules can be used to obtain the subsidiaries for the query name. By doing so, the named entities with more than one words will be parsed as one single constituent in the dependency tree. This will reduces parsing errors and help to find the correct slot values.

For *per:title*, *per:charges*, *org:political/religious affliation*, lists of trigger words are manually collected from the English Wikipedia. To avoid the typos in the texts, we first generate n-grams in the sentences. *n* can be from uni-gram to tri-gram determined by the number of trigger words used. After n-grams are generated, these n-grams will be compared with the triggers by means of the dice coefficient used in this paper.

As far as the types of *per:city_of_birth*, *per:cities_of_residence*, *org:city_of_headquarters* ect. are considered, the geo-name[9] lists are used to identify city, country and state. For the triggers of the slot *per:orgin*, we use the corresponding country name to find their demonym in Wikipedia.

1.9 Slot Filling Evaluation Results

We submitted one run for the slot filling task and the evaluation results of our system, the top two team and the median are shown in Table 3.

System	Precision	Recall	F-measure
Top-1 Team	35.03	25.5	29.52
Top-2 Team	49.17	12.59	20.05
Median Team	10.31	16.51	12.69
PolyU Team	**15.29**	**12.7**	**13.87**

Table 3: Results of the Slot Filling System

We only used the data provided by LDC for the slot filling task and the system has no access to the Internet during the evaluation. Result shows that our performance is slightly higher than the median team, but a long way from the top 2 systems. The reasons for low recall and precision are: (1) only sentences that contain the query name are considered as relevant. In actual text, however, slots might appear with no mention of the exact query name; (2) only top 50 documents related to the query name are used for extracting slot values. This is the primary reason why the recall of our system is much lower compared to those using 100 or more retrieved documents.

Conclusions and Future Work

This paper describes the entity linking and slot filling systems of the Hong Kong Polytechnic University team. For the entity linking, the system uses various resources and rank candidates based on Wikipedia context. The outcome on ranking is satisfactory, but the detection of NIL queries is still a problem. In the future, investigations will be conducted on finding suitable features to handle the NIL detection problem. For the additional NIL clustering system, our system is very simple. More features such as TF-IDF should be explored. Furthermore, as the context terms are modeled as Wikipedia terms, it is also possible to apply some network similarity measures such as the bipartition graph method (Tang, Lu, T. Wang, J. Wang, & W. Li, 2011).

Although the context modeling using Wikipedia anchors can achieve a high accuracy, it is rather time consuming to find high quality anchors from a span of text. And this method may probably fail in informal text. In the future, we will try to find more robust solutions with lower computation cost. The slot filling system combines the query expansion and pattern-based reasoning. Techniques like abbreviation extraction, name variation, and entity substitution are incorporated into this system. In the future, we can explore how to make use of sentences which did not have direct query mentions. Possible direction is to identify syntactic features for slot filling task such as adding co- reference resolution for named entities. Another possible direction is to classify sentences into suitable slot types based on training data first before extraction of information is conducted.

[9] http://www.geonames.org/

References

Castelli, V., Florian, R., & Han, D.-jung. (2010). Slot Filling through Statistical Processing and Inference Rules. *Proc. TAC 2010 Workshop*.

Chang, A. X., Spitkovsky, V. I., Yeh, E., Agirre, E., & Manning, C. D. (2010). Stanford-UBC Entity Linking at TAC-KBP. *Proceedings of the Third Text Analysis Conference (TAC 2010)* (Vol. 758).

Chen, Z., Tamang, S., Lee, A., Li, X., Lin, W.-pin, Snover, M., Artiles, J., et al. (2010). CUNY-BLENDER TAC-KBP2010 Entity Linking and Slot Filling System Description. *Proceedings of the Third Text Analysis Conference (TAC 2010)*.

Chrupala, G., Momtazi, S., Wiegand, M., Kazalski, S., Xu, F., Roth, B., Balahur, A., et al. (2010).

Saarland University Spoken Language Systems at the Slot Filling Task of TAC KBP 2010. *Proc. TAC 2010 Workshop*.

Cucerzan, S. (2007). Large-scale named entity disambiguation based on Wikipedia data. *Proceedings of EMNLP-CoNLL* (Vol. 2007, pp. 708–716).

Fern, N., Fisteus, J. A., S, L., & Mart, E. (2010). WebTLab□ : A cooccurrence-based approach to KBP 2010 Entity-Linking task. *Proceedings of the Third Text Analysis Conference (TAC 2010)*.

Lehmann, J., Monahan, S., Nezda, L., Jung, A., & Shi, Y. (2010). LCC Approaches to Knowledge Base Population at TAC 2010. *Proceedings of the Third Text Analysis Conference (TAC 2010)*.

Mcnamee, P. (2010). HLTCOE Efforts in Entity Linking at TAC KBP 2010. *Proceedings of the Third Text Analysis Conference (TAC 2010)*.

Milne, D., & Witten, I. H. (2008). Learning to link with wikipedia. *Proceeding of the 17th ACM conference on Information and knowledge mining - CIKM '08*, 509. New York, New York, USA: ACM Press. doi:10.1145/1458082.1458150

Radford, W., Hachey, B., Nothman, J., Honnibal, M., & Curran, J. R. (2010). Document-level Entity Linking□ : CMCRC at TAC 2010. *Proceedings of the Third Text Analysis Conference (TAC 2010)* (pp. 1-6).

Schwartz, A. S., & Hearst, M. A. (2003). A simple algorithm for identifying abbreviation definitions in biomedical text. *Pacific Symposium on Biocomputing* (Vol. 8, pp. 451–462). Citeseer.

Surdeanu, M., McClosky, D., Tibshirani, J., Bauer, J., Chang, A. X., Spitkovsky, V. I., & Manning, C. D. (2010). A Simple Distant Supervision Approach for the TAC-KBP Slot Filling Task. *Proc. TAC 2010 Workshop*.

Tang, J., Lu, Q., Wang, T., Wang, J., & Li, W. (2011). A Bipartite Graph Based Social Network Splicing Method for Person Name Disambiguation.

Varma, V., Bysani, P., Reddy, K., Reddy, V. B., Kovelamudi, S., Vaddepally, S. R., Nanduri, R., et al. (2010). IIIT Hyderabad in Guided Summarization and Knowledge Base Guided Summarization Track. *Proceedings of the Third Text Analysis Conference (TAC 2010)*.

[i] This work was done while they worked in HK Polytechnic University as exchange students.