NLPComp in TAC 2012 Entity Linking and Slot-Filling

Jian Xu* **Qin Lu*** **Jie Liu†** **Ruifeng Xu†**
Department of Computing, Hong Kong Polytechnic University, Hong Kong *
{csjxu, csluqin}@comp.polyu.edu.hk

Key Laboratory of Network Oriented Intelligent Computation,
Shenzhen Graduate School, Harbin Institute of Technology, China †
{lyjxcz, xuruifeng.hitsz}@gmail.com

Abstract

The NLPComp team participated in two TAC-KBP2012 tasks: Regular Entity Linking and Regular Slot Filling.

For the entity linking task, a three-step entity linking system is developed. In the first step, a list of possible candidates are 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 uses the Wikipedia anchor terms to enlarge the number of candidate instances. It then incorporates the topic modelling technique to select features of topical words. However, our system produces a poor answer coverage and the NIL detection system brings significant loss in the final F-score.

For the slot filling task, we developed a system which combines rule-based approach and multiple instance learning technique. In rule-based slot filling, a number of trigger words are collected from the English Wikipedia and the frequency feature is used to select final slot value(s). When extracting slot value(s) using the multiple instance learning technique, 2009 and 2010 KBP slot filling queries and manually annotated answers are used to create named entity pairs which exhibit a particular relation. Then bags of sentences containing named entity pair are extracted from the KBP source. Our system reaches the median level among all the participating systems.

1 Introduction

The regular entity linking task is to link a mention string to its corresponding Wikipedia entry, which is referred as the Knowledge Base (KB) node in the task. Besides participants are required to cluster mentions which do not appear in the KB.

Most of the previous 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 et al., 2010; Varma et al., 2010), Wikipedia redirects and disambiguation pages(Fern, Fisteus, Mart, 2010; Lehmann et al., 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. Besides, an entity can be selected as a candidate for a query if there is a name variant matching the query in its variant set created by leveraging the English Wikipedia (Zhang et al., 2011; Radford et al., 2011). Finding the acronyms for the named entity references or expanding acronyms were also used to generate candidates (Anastacio et al., 2011; Radford et al., 2011). The co-reference chain was used to find the canonical forms of the named entities (Radford et al., 2011).

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. Zhang et al. (2011) incorporated the surface features (surface match, word match, etc.), contextual features (bag-of-words, co-occurring named entities, etc.) and semantic features (named entity type and topic similarity) for candidate ranking. For the feature of topic similarity, they modeled the contexts as the probability distribution over Wikipedia categories. 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). Agirre et al. (2009) tried to use the machine learning technique to disambiguate the named entities with such features as anchor texts, lemmas in the spans, word/lemma/POS bigram and trigrams around the anchor text. Li et al. (2009) employed a Listwise *Learning to Rank* model and augmenting Naive Bayes model to rank the candidate. Zhang et al. (2010) proposed a system of using Lucene-based ranking, SVM-rank and binary SVM classifier for entity linking.

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.

To extract slot value(s) using supervised learning technique, the distant supervision algorithm is mostly used (Agirre et al., 2009; Surdeanu et al., 2010; Sun et al., 2011). Some researchers have combined the rule-based approach and supervised machine learning method (Gao et al., 2010; Chada et al., 2010), hybrid approaches (Chen et al., 2010; Castelli et al., 2010).

In this paper, our slot filling system incorporates rule-based approach and multiple instance learning technique 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. In rule-based slot filling, a number of trigger words are collected from the English Wikipedia and the frequency feature is used to select final slot value(s). When extracting slot value(s) using the multiple instance learning technique, 2009 and 2010 KBP slot filling queries and manually annotated answers are used to create named entity pairs which exhibit a particular relation. Then bags of sentences containing named entity pair are extracted from the KBP source.

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.

2 The Entity Linking System

Our entity linking system has two components: one is the candidate generation and the other is the candidate selection. In the second component of

the candidate selection, *NIL* responses are required to be clustered. Thus we included the hierarchical clustering approach to handle the *NIL* queries.

2.1 Knowledge Base preprocessing and preparation

To fully utilize the linking information in the English Wikipedia, a 2011 Wikipedia dump [1] is downloaded. Hereafter, the Wikipedia Miner Toolkit [2] is used to managing the Wikipedia resources. Lucence [3] is used to index the KB nodes with the fields including *type*, *name*, *title*, *text*, *infoBox*, *alternate name*. The source documents were also indexed to find the some required document quickly.

2.2 Query expansion

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

(1) Abbreviated tokens in organization names are replaced by their full forms, for example,

$$Ltd./ltd. \Rightarrow Limited$$
$$Inc./inc. \Rightarrow Incorporation$$
$$Corp./corp. \Rightarrow Corporation$$

(2) Names in the background document are extracted to expand queries.

The Stanford named entity recognizer [4] 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 String kernel (Lodhi et al., 2002) to measure the similarity between the person name and query name, and only keep those person names whose similarities are greater than the given threshold.

(3) Use the abbreviation extraction technique to find the full names of acronyms from the background document

To get the full expressions of the 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. In the entity linking and slot filling tasks, 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. Take the query *BNA* for example:

$$BNA \Rightarrow Bahrain\ News\ Agency$$

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 *Agency* of the long form. If A is not found in *Agency*, this algorithm will move to the next word *News* before *Agency*, 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 *Agency*, this algorithm will move to the next letter N and next word *News*, and check if N is in *News*. 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.

2.3 Candidate generation

The candidate generation part of our system generates a candidate entities set for each query. Several approaches are used to generate candidates. They are:

A1. Match the query with the entities. Our system selected the top 10 searching results got with the Lucence of each query as the candidates.

A2. Match the query with the candidates' alternate names. Some queries have no canonical forms. They might be the alternate names of

[1] http://sourceforge.net/projects/wikipedia-miner/files/data/en/enwiki-20110722-csv.tar.gz/download
[2] http://wikipedia-miner.cms.waikato.ac.nz/
[3] http://lucene.apache.org/
[4] http://nlp.stanford.edu/

candidate terms. So, in our system, the top 100 searching results are returned using Lucence and select the searching results' alternate names as candidates.

A3. Use the SFEM. First of all, all possible *senses* were found using the Wikipedia Miner tool. 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. The entities matching the surface form as some page title and its redirects page titles were selected as candidates.

A4. Expand the acronyms. For the acronym queries, the system tried to get their canonical forms. A simple algorithm (Schwartz, 2003) is used to find the abbreviation definitions.

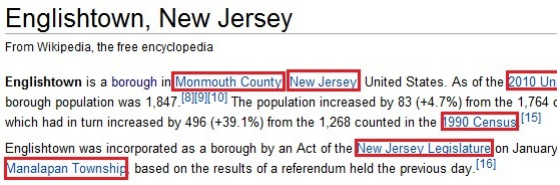
Finally, for each query, our system would select the candidates generated from step **A1** to **A4** and remove repetitive entries in the lists.

2.4 Candidate selection

In Section 2.3, a lot of noise candidates will be generated. Hence, the purpose of candidate selection is to filter out these irrelevant candidate entities. In this task, we took two approaches to select the most likely candidate for the query. One is the supervised approach using the multi-class SVM algorithm and the other is maximizing the similarity between query background document and candidate documents. Before selecting candidate for a query, we need to expand candidate instance and to create features for the two candidate selection approaches. Candidate instances are expanded with anchor terms embedded in their Wikipedia articles. The features for the two approaches are topical words sampled from the query background document and candidate documents.

2.4.1 Expand candidate instances with anchored terms

Given a query, each candidate has only one document in the KBP reference. To have more examples describing the candidate, we extracted the anchored terms in the candidate article. Take the query *Englishtown* for example, we have its candidate *Englishtown, New Jersey* and this candidate article in Wikipedia is shown as follows:


Englishtown, New Jersey

From Wikipedia, the free encyclopedia

Englishtown is a borough in Monmouth County, New Jersey, United States. As of the 2010 Un
borough population was 1,847.[8][9][10] The population increased by 83 (+4.7%) from the 1,764 c
which had in turn increased by 496 (+39.1%) from the 1,268 counted in the 1990 Census.[15]

Englishtown was incorporated as a borough by an Act of the New Jersey Legislature on January
Manalapan Township, based on the results of a referendum held the previous day.[16]

Figure 1: Anchor terms in Wikipedia

From the *Englishtown, New Jersey* article, anchor terms such as *Monmouth County, New Jersey*, *1990 Census*, *New Jersey Legislature*, *Manalapan Township* can be extracted. Since most of anchor terms have their own articles in Wikipedia, we then use articles of these anchor terms to expand candidate instance. After expansion, these articles share the same class label with the candidate instance. In this sense, the number of instances for the candidate class are increased and the number of candidate documents will be increased as well.

2.4.2 Feature selection using topic modelling technique

We assume that documents that have similar topics are expected to have similar topic words. Take the query *Santa Cruz* for example, after the topic sampling from the background document and candidate documents, we obtained topical words under each topic listed in **Figure 2**.

1st topic	2nd topic	3rd topic	4th topic
colombium	japanese	japanese	band
colombian	force	japan	album
department	carrier	war	song
festival	aircraft	yamamoto	version
national	battle	american	single
senate	u.s.	imperial	merengue
president	naval	fleet	music
constitution	attack	army	first
caribbean	ship	empire	dominican
percent	island	emperor	artist

Figure 2: Four topics for the *Santa Cruz* query

Obviously, the first is the *politics* topic and the second and third are about the *war* topic and the fourth is related to the *music* topic. We then use these topical words as features and

compute similarity between background document and candidate documents based on these topic words.

Now remains the question of how obtain these topical words. In this paper, we conduct the topic modeling through the LDA (Blei et al., 2003). LDA is a generative model which is based on probabilistic sampling techniques investigating how words in documents are generated with the hidden variables (Steyvers and Griffiths, 2006). It models observations using a set of component distributions (Heinrich, 2005). The main idea of LDA is to model documents in terms of topics where a topic is defined as a distribution over a fixed vocabulary of words. If K topics are mined from a collection of documents, then each document will K topics with different distributions.

In the LDA model, words in documents are observable and topics are latent variables hidden in these documents. Its graphical representation is depicted as,

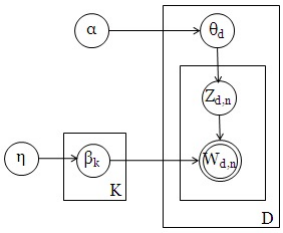


Figure 3: LDA graphical representation

In the **Figure 3**, each node denotes a random variable and the edge between nodes represents dependency relations between nodes. The double circle around the random variable denotes an observable node (evidence node). The plate surrounding the nodes indicates N i.i.d samples. D and K refer to the number of documents and the number of topics, respectively. α and η are hype-parameters on the mixture proportions for topics and documents. θ_d refers to the multinomial topic distributions for document d and β_k is multinomial word distributions for topic k. $Z_{d,n}$ denotes a topic from which the n^{th} word in document d is drawn and $W_{d,n}$ indicates the observable n^{th} word in document d. In the LDA model, for a document d, a vector of topic distributions $\overline{\theta}_d$ is drawn from a Dirchlet distribution $\overline{\theta}_d \sim Dir(\overline{\alpha})$; topic assignment for n^{th} word $Z_{d,n}$ follows from a multinomial distribution $Z_{d,n} \sim Mult(\overline{\theta}_d)$; and the n^{th} word in document d is sampled from multinomial distribution $W_{d,n} \sim Mult(\overline{\beta}_{Z_{d,n}})$.

After obtaining the topical words from each topic, we then compute the *TF*IDF* scores of topical words in the background document and candidate documents. In this representation of documents, each topical word corresponds to a feature with the *TF*IDF* score as its value.

2.4.3 Candidate selection using multi-class SVM

To select the most likely candidate for a query, we apply the multi-class SVM approach. Traditionally, the multi-class classification problem can be decomposed into binary classification tasks. And the commonly used strategies include One-versus-all (OVA) and One-versus-one (OVO) (Aly, 2005; Milgram et al., 2006). The OVA strategy solves the multi-class problem by building one SVM for each class, which is trained to discriminate the samples in a given class from the samples in all the other classes. When classifying a new instance, the classifier with the maximum output will be chosen and the corresponding class label will be given to the new instance. The OVO strategy builds one SVM for each pair of classes. If we have M classes, we will have to build $\frac{M(M-1)}{2}$ binary classifiers. When classifying a new instance, a voting technique is employed to select the class that has the maximum votes. In the entity linking task, we have used the OVO strategy for selecting the most likely candidate.

2.4.4 Candidate selection by maximizing similarity

In this approach, a vectorial representation of query background document is compared with the vectorial representations of the candidate documents. The features used in these vectorial representations are topical words with assigned *TF*IDF* scores. We then choose the candidate that has the maximum similarity with query background document, defined as,

$$\arg max \cos(\boldsymbol{q}, \boldsymbol{c}_i), i \in [1, n] \quad (1)$$

where i is the i^{th} candidate among the n candidates for the query q. The cosine similarity

between the query background document vector $\boldsymbol{q}$ and the candidate document vector $\boldsymbol{c}_i$ is defined as,

$$\cos(\boldsymbol{q}, \boldsymbol{c}_i) = \frac{\boldsymbol{q} \cdot \boldsymbol{c}_i}{\|\boldsymbol{q}\|\|\boldsymbol{c}_i\|} \quad (2)$$

Using the equation (1) we can select the candidate that has the maximum similarity with the query background document and link this candidate to the specific query.

2.5 NIL query clustering system

In KBP 2012 entity linking task, the *NIL* queries are required to be clustered based on the underlying entities. Each *NIL* query is represented by the topical words and cosine metric is taken to compute similarity between queries. Then, a simple Hierarchical Agglomerative Clustering (HAC) algorithm is used to cluster the queries.

2.6 Evaluation on entity linking system

Only two runs are submitted. The first run, denoted as *RUN1*, uses the multi-class SVM approach to select a most likely candidate for a query. The second run, denoted as *RUN2*, uses the maximum similarity approach to get a candidate that shares the maximum similarity with a query's background document.

In *RUN1* experiments, the JGibbLDA [5] tool is used to get the topical words in documents. The values for the hyper-parameters α and β are $\frac{50}{K}$ (K is the number of topics) and 0.01 (Steyvers and Griffiths 2006) and the number of iterations is set to 500. Note that the query background document and candidate documents are placed together to obtain the topical words for each document. In this task, the top 50 topical words under each topic are used. In terms of classifier, the LIBSVM [6] tool with one-versus-one strategy is used and default parameters are kept. Then the performances of two runs are listed in **Table 1**.

	micro-avg.	B3+prec.	B3+rec.	B3+F1
RUN1	0.129	0.093	0.121	0.106
RUN2	0.306	0.261	0.289	0.274

Table 1: Evaluation of entity linking system

[5]http://jgibblda.sourceforge.net/

[6]http://www.csie.ntu.edu.tw/ cjlin/libsvm/

Given that the highest F1 score among all participants is 0.73 and the median F1 score is 0.536, the system performance is far below the median level. In the analysis, we found that our system has a poor answer coverage and it only has 41.15% coverage of the answers. Worse still, the NIL detection system has a poor performance since it has only generated 76 *NIL* queries while the manual answer has 1049 *NIL* queries. Besides, using the anchor terms in the candidate text would introduce noise as well, since the anchor terms in the candidate documents may share no topical words with the candidates, thus bringing noises into the topic modeling process. On the other hand, the number of topics is crucial to the topic modeling. In this task, we simply set the number of topics to the size of candidates. If the number of topics is known beforehand, the latent topics can be well modeled out of the document collections by LDA. In fact, the number of topics is expected to be estimated using such approaches as cross validation, the nonparametric Bayesian method of Hierarchical Dirchlet processes (Teh et al., 2006).

3 The Slot Filling System

Our slot filling system has three modules. The first is for document retrieval and preprocessing. The second is for rule-based slot-filling and third is for slot-filling with multiple instance learning.

3.1 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 including the named entity recognizer tool and part of speech tagger to preprocess the documents. We also used the OPENNLP tokenizer and sentence detector [7] to tokenize the documents and find the sentence boundaries of the documents.

3.2 Rule-based slot filling

For a particular set of slots, *per:title*, *per:charges*, *per:religion*, *org:political/religious affliation* and so on, lists of trigger words are used. These trigger words are manually collected from the English

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

Wikipedia. Example trigger words are listed in **Figure 4**.

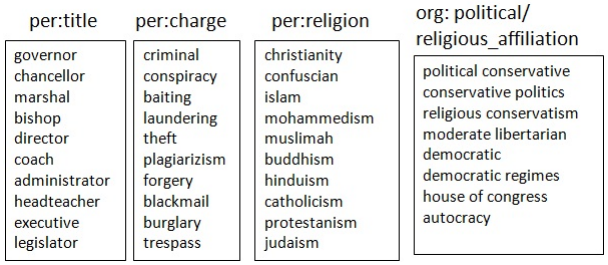


Figure 4: Trigger words

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 for the types of *per:date of birth*, *org:founded*, *org:dissolved*, etc., 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 [8] lists to identify city, country and state.

After obtaining slot values for a query name, we rank these slot values by the number of times they co-occur with the query name. For single valued slots, a slot value with the highest frequency will be returned. For the list-valued slots, the top 5 slot values will be returned.

3.3 Slot filling with multiple instance learning

For the types of *per:parents* & *per:children*, *org:parents* & *org:subsidiaries*, we used the multiple instance learning approach to extract slot values. Given a number of pairs of named entities known to have some kind of relation, bags of these sentences containing the named entity pairs are extracted from the KBP source documents. For example, *Chris Dodd* is father of *Grace*, we have the following two sentences describing this relation,

Cox News Service WASHINGTON - On a sunny April day in 2006 , Democratic Sen. ***Chris Dodd*** *asked his wife to take a walk beside the Connecticut River near their home in East Haddam . I said , 'Great , I 'll get the girls ready , ' but he said , ' I think we need a baby sitter , ' recalled Jackie Clegg Dodd . At the time , their daughter* ***Grace*** *was 4 and her sister Christina was 1 .*

Grace *and Christina love to ring the bell by pulling the rope that comes down through the ceiling , their mom said . After the school closed , the structure became a rehearsal hall for the Goodspeed Opera House – where Man of La Mancha debuted in 1964 , recalled Jackie Dodd . So it was at Sen.* ***Chris Dodd*** *'s house in Connecticut where they perfected the song : To Dream the Impossible Dream.*

In this task, intermediate sentences are allowed in-between named entity pairs. These named entity pairs are extracted from 2009 and 2010 slot filling evaluation queries and manually annotated answers. With these named entity pairs, we start to extract sentences which contain the named entity pairs from the KBP source. We then address the task of extracting slot values as a multi-instance learning (MIL) (Zhou, 2004; Bunescu and Mooney, 2007). In MIL, the training class labels are associated with set of bags, instead of individual class labels or patterns. Also, a bag is positive if this bag contains at least one positive pattern.

Given a training set χ, let χ_p be set of positive bags and χ_n the set of negative bags, $\chi_p \in \chi$ and $\chi_n \in \chi$, then the multiple instance SVM can be formalized as minimizing the object function $\jmath_{obj}$ defined by,

$$\jmath_{obj}(w, b, \xi) = \frac{1}{2}\|w\|^2 + \frac{C}{L}(c_p \frac{L_n}{L}\Xi_p + c_n \frac{L_p}{L}\Xi_n) \quad (3)$$

where Ξ_p, Ξ_n, L_p and L_n are defined as,

$$\Xi_p = \sum_{X \in \chi_p} \sum_{x \in X} \xi_x \quad (4)$$

$$L_p = \sum_{X \in \chi_p} |X| \quad (5)$$

$$\Xi_n = \sum_{X \in \chi_n} \sum_{x \in X} \xi_x \quad (6)$$

$$L_n = \sum_{X \in \chi_n} |X| \quad (7)$$

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

subject to:

$$w\phi(x) + b \geq +1 - \xi_x, \forall x \in X \in \chi_p \quad (8)$$

$$w\phi(x) + b \leq -1 + \xi_x, \forall x \in X \in \chi_n \quad (9)$$

$$\xi_x \geq 0$$

The capacity control parameter C is divided by L, the summation of positive instance and negative instances, $L_p + L_n$. The positive parameter c_p controls the relative influence that false negative and false positive errors have on the object function in (3) (Bunescu and Mooney, 2007). Additionally, SVM is well known for using kernel methods to handle non-separable data points by the hype-plane in the input space. They are able to represent linear patterns in high-dimensional space (Shawe-Taylor and Cristianimi, 2004). In this task, we used the radial basis function (RBF) kernel.

3.4 Slot filling evaluation results

To evaluate the performance of the slot filling system, some slots are filled using rule-based approach. This approach is quite straightforward. The number of trigger words used in this task is over 2500. In terms of the multiple instance learning approach for slot filling, the 2009 and 2010 evaluation queries and manually annotated answers are used to extract sentences which contain named entity pairs. Each named entity pair is expected to exhibit some sort of relation, that is, the slot type, for example, *per:children*, *per:parents*, *org:parents*, *org:subsidiaries* and so forth. For each named entity pair, intermediary sentences are allowed. From these sentences, tokens are extracted as features. For the sake of simplicity, only binary value is assigned to a feature. If a feature is in a sentence that contains a named entity pair, then 1 is assigned to this feature, otherwise 0 is set to it. Based on this feature representation, we applied the multiple instance learning algorithm on the 2012 evaluation queries.

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 2**.

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 the same as the median team in F1 metric, 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; (3) in the multiple instance learning framework, the number of sentences in a bag is restricted by the KBP source files.

	precision	recall	F1
Top-1 Team	0.68	0.42	0.51
Top-2 Team	0.49	0.21	0.3
Median Team	0.11	0.09	0.1
NLPComp	0.12	0.08	0.1

Table 2: Evaluation of the slot filling system

4 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 topical words sampled from query background document and candidate documents. The poor answer coverage and the detection of NIL queries brings a great loss in F1 measure. In the future, investigations will be conducted on finding suitable approaches to increase the answer coverage and 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 et al., 2011).

The slot filling system combines rule-based approach and multiple instance learning technique. Techniques like abbreviation extraction, name variation, and string kernel 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. In multiple instance learning framework, the number of sentences for a particular bag is limited by the KBP source and we plan to use named entity pairs as search queries and then extract sentences contained named entity pairs from the Internet, which can leverage the vast amount of information available on the web.

References

Angel X. Chang, Valentin I. Spitkovsky, E. Yeh, E. Agirre and Christopher D.Manning. 2010. Stanford-UBC Entity Linking at TAC-KBP. In *Proceedings of the Third Text Analysis Conference (TAC 2010)*.

A. S. Schwartz, M. A. Hearst. 2003. A simple algorithm for identifying abbreviation definitions in biomedical text In *acific Symposium on Biocomputing*, pp 451-462.

A. Sun, R. Grishman, W. Xu, and B. Min. 2011. New York University 2011 System for KBP Slot Filling In *Proceedings of the Fourth Text Analysis Conference (TAC 2011)*.

D. Blei, A. Ng and M. Jordan 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993V1022.

D. Milne, I. H. Witten. 2008. Learning to link with wikipedia. In *Proceeding of the 17th ACM conference on Information and knowledge mining - CIKM 08*, New York, New York.

D. Chada, C. Aranha and C. Monte 2010. An Analysis of The Cortex Method at TAC 2010 KBP Slot-Filling. In *Proceedings of the Third Text Analysis Conference (TAC 2010)*.

G. Chrupala, S. Momtazi, M. Wiegand, S. Kazalski, F. Xu, B. Roth, A. Balahur, et al. 2010. Saarland University Spoken Language Systems at the Slot Filling Task of TAC KBP 2010. In *Proc. TAC 2010 Workshop*.

G. Heinrich 2005. Parameter Estimation for Text Analysis. Technical report.

H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, C. Watkins. 2002. Text classification using string kernels. *The Journal of Machine Learning Research*, pp. 419-444.

I. Anastacio, B. Martins, P. Calado, et al. 2011. Supervised Learning for Linking Named Entities to Knowledge Base Entries. In *Proceedings of the Fourth Text Analysis Conference (TAC 2011)*.

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

J. Milgram, M. Cheriet and R. Sabourin. 2006. One Against One or One Against All: Which One is Better for Handwriting Recognition with SVMs? In *10th International Workshop on Frontiers in Handwriting Recognition (IWFHR 2006)*.

J. Shawe-Taylor and N. Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.

J. Tang, Q. Lu, T. Wang, J. Wang, and W. Li. 2011. A Bipartite Graph Based Social Network Splicing Method for Person Name Disambiguation. In *SIGIR 2011*, pp. 1233-1234.

M. Aly. 2005. Survey on Multi-Class Classification Methods. Technical Report, Caltech, USA.

M. Surdeanu, D. McClosky, J. Tibshirani, J. Bauer, A. X. Chang, V. I. Spitkovsky and C. D. Manning. 2010. A Simple Distant Supervision Approach for the TAC-KBP Slot Filling Task. In *Proceedings of the Third Text Analysis Conference (TAC 2010)*.

M. Steyvers and T. Griffiths. 2006. Probabilistic topic models. In *Latent Semantic Analysis: A Road to Meaning*, Laurence Erlbaum.

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

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

Razvan C. Bunescu and Raymond J. Mooney. 2007. Learning to Extract Relations from the Web using Minimal Supervision.

S. Cucerzan. 2007. Large-scale named entity disambiguation based on Wikipedia data. In *Proceedings of EMNLP-CoNLL*, pp. 708V716.

S. Gao, Y. Cai, S. Li, Z. Zhang, J. Guan, Y. Li, H. Zhang, W. Xu and J. Guo 2010. PRIS at TAC2010 KBP Track. In *Proceedings of the Third Text Analysis Conference (TAC 2010)*.

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

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

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

W. Zhang, Chew L. Tan, J. Su, B. Chen, et al. 2011. I2R-NUS-MSRA at TAC 2011: Entity Linking. In *Proceedings of the Fourth Text Analysis Conference (TAC 2011)*.

W. Radford, J. Nothman, J. R. Curran, B. Hachey, M. Honnibal. 2011. Naive but effective NIL clustering baselines - CMCRC at TAC 2011. In *Proceedings of the Fourth Text Analysis Conference (TAC 2011)*.

Y. Teh, D. Newman and M. Welling. 2006. A collapsed variational Bayesian inference algorithm for latent Dirichlet allocation. In *Neural Information Processing Systems*.

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

Z.-H. Zhou. 2004. Multi-instance learning: A survey. Technical Report, Nanjing University, Nanjing, China.