The CASIA Entity linking System at TAC 2013

Yubo chen,Guangyou zhou,Liheng Xu,Shizhu He,Kang Liu and Jun Zhao
National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Sciences
HaiDian District, Beijing, China.
{yubo.chen,gyzhou,lhxu,shizhu.he,kliu,jzhao}@nlpr.ia.ac.cn

Abstract

In this paper, we describe our entity linking system at TAC-KBP 2013. Our system consists of four modules. 1) Query expansion module. 2) Candidate generation module. 3) Candidate Entity disambiguation module. 4) NIL clustering module. First, we expand the queries with the information of the query documents. Then we find the candidates of queries from the Knowledge Base using the WikiPedia knowledge and string matching method. In the disambiguation module, we select a candidate entity using a supervised learning to rank method based on a variety of evidences. Finally, we cluster all the NIL entities which do not been detected in the above modules using the Hierarchical Agglomerative Clustering algorithm. The evaluation results show that our method is effective for English entity linking task.

1 Introduction

The CASIA team participated in the English entity linking task in the KBP track of TAC 2013. The goal of Knowledge Base Population (KBP) track at Text Analysis Conference (TAC) 2013 is to automatically discover information about named entities and link this information in a reference knowledge base.

Following the literature (Ji et al., 2010), the task of entity linking is defined as follows: given a query that consists of a name string, a background document ID, and a pair of UTF-8 character offsets indicating the beginning and the end locations of the name string in the document, the system is required to provide the ID of the KB entry to which the name refers, or a "NILxxxx" ID if there is no such KB entry. The entity linking system is required to cluster queries referring to the same non-KB (NIL) entities and provide a unique ID for each cluster, in the form of "NILxxxx" (e.g., "NIL0021"). For example, given the following two queries which contain the name string "ABC":

1) ABC could not directly access the underlying file system and operating system.

2) In American, ABC first broadcast on television in 1948.

An Entity Linking system should link the two ABC mentions with the real world entity American Broadcasting Company and ABC (programming language), respectively.

Entity linking task is challenging due to name variations and entity ambiguity (Shen et al., 2012). In reality, an entity may have multiple surface forms. For example, the IBM Company can be mentioned as IBM, Big Blue and International Business Machine, which respectively correspond to its abbreviation, nickname and its Official Title. Besides, one entity mention may also refer to several different real world entities. For example, the entity mention AI can refer to more than 10 real world entities, such as Artificial intelligence, Game artificial intelligence, the Singer Ai, etc.

To solve the above two problems, we propose an entity linking system consisting of the following four modules. 1) Query expansion module: We expand the queries from the background documents; 2) Candidate generation module: With the query generated in the first step, this module retrieves the candidates from the Knowledge Base using the Wikipedia

knowledge and a string matching method; 3) Candidate Entity disambiguation module: Our system selects a single entity using a supervised learning-to-rank method; 4) NIL clustering module: We cluster the queries referring to the same non-KB (NIL) entities using the Hierarchical Agglomerative Clustering algorithm.

The rest of this paper is organized as follows. We first describe how to expand the query in Section 2. Section 3 introduces the candidate selection stage of our system. The entity disambiguation module is presented in Section 4. Section 5 presents our clustering method. The experiment results are presented in Section 6. Finally,we conclude this paper in Section 7.

2 Query Expansion

This module mainly addresses the acronym problem, we expand the queries from the background documents using the heuristic rules. From our observation, acronym query name string is usually high ambiguous, but its full name is usually unambiguous. Thus, expanding the query from its document can effectively reduce the ambiguities of the query. For example:

1) The ABC (Australian Broadcasting Corporation) is Australia's national public broadcaster.

2) In American, ABC (American Broadcasting Company) first broadcast on television in 1948.

If we expand the entity mention "ABC" from the background documents we can easily obtain the full name "Australian Broadcasting Corporation" and "American Broadcasting Company" which are more unambiguous in the next process. In this paper, we identify the abbreviations of the entities using the heuristic rules (Zhang et al., 2012). First, for the capitalized query C, we check whether the document contains pattern (C) or not. If the document contains the pattern (C), we extract the continuous sequence of n tokens that start with the first letters of the acronym as the target entity, where n represents the number of the letter in C. Besides, we also observe that some queries usually contain the part of its full name, thus if a query is wholly contained in a string of a named entity in the associated document, we use the named entity as the full name of the query.

3 Candidate Generation

In this module, we generate the set of candidate entities for each expanded query. Intuitively, the candidate entities of the query should have the same or similar surface form of the entity mention or refer to the same entity with the entity mention. In this paper, we find the candidate entity using the Wikipedia knowledge and the string matching method.

3.1 Wikipedia Knowledge

We first take advantage of the huge amount of knowledge available in Wikipedia. The structure of Wikipedia provides a set of useful features for the generation of the candidate entities. In this paper, we generate the candidate entities using the redirect pages, disambiguation pages and anchor texts in Wikipedia article:

- **Entity pages:** Each entity page in Wikipedia describes a single entity and contains the background information focusing on this entity. Generally, the title of each page is the most common name for the entity described in this page. We select the entity as the candidate entity if the entity mention is an exact match with the title.

- **Redirect pages:** A redirect page is an aid for navigation. So redirect pages often indicate synonym terms, abbreviations or other variations of the pointed entities. Thus, we select the entity as the candidate entity if the entity mention is exactly match of the alternative name of the entity. For example, "ABC (American television)" represent the same entity with the "American Broadcasting Company", the page "American Broadcasting Company" was redirected from "ABC (American television)". So for the entity mention "ABC (American television)", the entity "American Broadcasting Company" should be a candidate entity.

- **Disambiguation pages:** In Wikipedia, when multiple entities in Wikipedia could be given the same name, a disambiguation page is created to separate them. It contains a list of references to the pages for these ambiguous entities that share the same name. So if the entity

mention matches the title of the disambiguation page or any one name of the listed articles, we add all the entities listed in the disambiguation page to the candidate set.

- **Anchor text:** The anchor text is the hyperlink contained in the article, which links to the pages of entities mentioned in this article. So the anchor text provides a very useful source of synonyms and other variations of the entity, and can be regarded as the surface form of that linked entity. In this module, if the entity mention matches the anchor, we add all the anchor target entities into the candidate set.

3.2 String Match

Though with use of the Wikipedia knowledge we can recall the majority of the right answers, there are still some entity mentions for which we cannot find the right candidate entity. The main reason is that the entity mention for this query is misspelled. To solve this problem, we select the entity which has a high edit distance with the entity mention as the candidate entity. For example, the edit distance between "American Broadcasting Corporation" and "American Broadcasting Company" is 0.78, thus we can select the entity "American Broadcasting Corporation" as the candidate of the "American Broadcasting Company". In this module, we set the threshold to 0.6 for the edit distance empirically. Though the threshold could be tuned to minimize the candidate set and maximize the recall, we will leave it for our future work.

To evaluate the candidate generation module, we evaluate the coverage of the candidate entity set in TAC-KBP track 2010 data, TAC-KBP track 2011 data and TAC-KBP track 2013 data . In table 1, we show the recall of the candidate entity in the three data sets.

Data Set	TAC 2010	TAC 2011	TAC 2013
Recall	0.9029	0.7580	0.8110

Table 1: Recall of the candidate generation.

4 Candidate Entity Disambiguation

To select a single entity which has the most probability to be right answer, we use a supervised learning-to-rank model in this phase. First, we represent each entity mention and the associated candidate entity as a series of features representing contextual, semantic, surface and NIL evidence. Then in the ranking module, we give each candidate entity a score based on these features. Finally, we select the entity with the highest score as the answer. The intuition behind this is that the correct entity should receive a higher score than all other entities by using the ranking SVM algorithm (Joachims, 2002). Here, we use the linear kernel due to the efficiency compared with other kernel functions, and set the slack parameter C to 20. Furthermore, we take the loss function as the total number of swapped pairs summed over all training examples. In the following we will describe the features used in the ranking model.

4.1 Surface Features

The features in Surface group are used to measure the similarity between the name string of the query and name string of the candidate entities. For the candidates containing the parenthetical expression, we compute their similarity after removing the parenthesis. For example, the similarity between "American Broadcasting Company(TV)" and "American Broadcasting Company" is 1. For the NIL entity, this feature is set to 0. Besides, we also take the effect of the acronym and the case of the string into account (e.g., the similarity between the "abc (broadcasting)" and the "American Broadcasting Company" is also 1). Finally, we compute the similarity between the candidate entity and the entity mention using the following two methods:

- **Edit Distance Similarity:** The feature value is the edit similarity between the mention string and the candidate. In our system the edit distance similarity is calculated using the formula 1:

$$edit_{a,b}(i,j) = \begin{cases} max(i,j) & if min(i,j) = 0, \\ min \begin{cases} edit_{a,b}(i-1,j)+1 \\ edit_{a,b}(i,j-1)+1 \\ edit_{a,b}(i-1,j-1)+[a_i \neq b_j] \end{cases} & otherwise \end{cases} \quad (1)$$

Note that the first element in the minimum corresponds to deletion (from a to b), the second to insertion and the third to match or mismatch, depending on whether the respective symbols are the same.

- **Dice Coefficient:** We compute the dice coefficient score between the entity mention and the title of the candidate entity as a feature. For example, x and y are two strings, the coefficient may be calculated as the formula 2.

$$S = \frac{2n_t}{n_x + n_y} \quad (2)$$

where n_t is the number of character bigrams found in both strings, n_x is the number of bigrams in string x and n_y is the number of bigrams in string y.

4.2 Contextual Features

The features in Context group are used to measure the contextual co-occurrence between the query source document and the KB definition of the candidate entities. The intuition behind this is that if the query name and the candidate entity refer to the same entity, they may be used in the similar contexts. We mainly consider the similarity based on VSM model and the entity co-occurrence.

- **Similarity based on VSM model:** This feature captures the similarity between the document of the query and the document of the candidate entity based on the VSM model. To calculate the similarity, we represent the candidate entity and the query as the bag-of-word vector, each entry in the vector is weighted using the standard TF-IDF measure. Finally, we calculate the cosine similarity between vectors as the feature. For NIL entity, the value is set to 0.

- **Entity co-occurrence:** This feature marks the presence of names in the text. Two features are included, one feature is whether the query name string appears in the KB text, the other is whether the title of the candidate entity appears in the source document of the query. For the NIL entity, both of the feature values are set to 0.

4.3 Semantic Features

The features in semantic group are used to measure the relatedness between the query source document and the KB definition of the candidate entities. The contextual features can only capture the word or entity co-occurrence information, which is usually not enough for our task. For example, If two entities represent the same real entity, but they share no word or rare word co-occurrence, it's difficult to distinguish the two entities, so we should take the semantic relatedness into consideration. In our system, we mainly use the following features to calculate the semantic features.

- **Entity Popularity:** This feature tells us the likelihood of an entity appearing in a document. We calculate it in the same way with (Han and Sun, 2011). In the system, the entity popularity can provide a priori information to the possible referent entities of a name mention. For example, given the entity "Michael Jordan", without any other information, we can know the popularity of the entity basketball player "Michael Jeffrey Jordan" is higher than that of the entity Berkeley professor "Michael I. Jordan". So we use it as a feature of our system. For NIL entity the value is 0.

- **Semantic Relatedness of Wiki-Concept:** In our system the semantic relatedness of Wikipedia contains two aspects. One is the average semantic relatedness between the query name and each concept in the KB definition of the candidate entities. The other is the average semantic relatedness between the candidate entity's title and each concept in the source document of the query. Our system calculate the Semantic Relatedness Of Wikipedia Concept in 2 steps. First, we recognize the Wikipedia concept in both of the query source document and the KB definition of the candidate entity, using the Wikipedia-Miner toolkit developed by (Milne and Witten, 2012). Then we calculate average semantic relatedness between the query name and each concept in the KB definition of candidate entity. And we also get the average semantic relatedness between the candidate entity's title and each concept in the source document of the query.
  Like (Witten and Milne, 2008) (Milne and Witten, 2008),We calculate the relatedness score between two concepts as follows:

$$sr(c_i, c_j) = 1 - \frac{\log(max(|C_i|, |C_j|)) - \log(|C_i \cap C_j|)}{\log(W) - \log(min(|C_i|, |C_j|))} \quad (3)$$

where W is a set of all Wikipedia articles, c_i and c_j are two Wikipedia articles, C_i and C_j are the sets of all articles that link to c_i and c_j, respectively. For the NIL entity, the value is set to 0.

4.4 NIL Features

Following the literature of (Dredze et al., 2010), we add a special entity NIL into the candidate entity. We also add a NIL feature to the feature set. So we can decide whether the entity mention link to the NIL or not. In our system, we set the best value of the NIL feature to 0.7 on the development set.

5 NIL clustering

After the above module, we link many entity mention to the NIL entity. The task require the system to cluster together queries referring to the same entities and provide a unique ID for each cluster. So in the NIL clustering module, we take all the queries which link to the NIL entities as input and get the clusters with the unique ID. First, we cluster the NIL queries based on their surface forms. In this step, we mainly use the heuristic rules. If the edit distance between two entity mention is higher than 0.6, we classify them in the same cluster. In this step the queries whose entity mentions contain the other are also believed to belong to the same cluster. After the rough classification, we use hierarchical agglomerative clustering (HAC) algorithm to cluster NIL queries in each cluster obtained from the first step. The second step works as follows: Initially, each query is an individual cluster; then we iteratively merge the two clusters with the largest similarity to form a new cluster until this similarity is smaller than a threshold. In this algorithm we use the average link method to compute the similarity between two clusters. The similarity between queries is determined by the query source document similarity based on VSM model.

6 Experiments and Results

The KBP 2010 training data set is used as our training data. This data set contains 1500 queries which are selected from the English newswire articles and the web pages. We use the KBP 2011 evaluation data set as the development data. The KBP 2011 evaluation data set contains 2250 queries which are selected from English newswire articles and web pages.We use the KBP 2013 English entity linking task contains 2190 queries as the test data, this data are selected from English newswire articles, web pages and discussion forum documents. In the task, we use the Bˆ3+F1 to evaluate the results by using the score tool[1]. We submit the 5 runs of our system shown in Table 2. The methods highest score is 0.573, and each parts score is shown in Table 3.

Submission	Bˆ3+F1
CASIA1	0.535
CASIA2	0.536
CASIA3	0.534
CASIA4	0.549
CASIA5	0.573

Table 2: Submissions scores

Evaluation metric	Bˆ3+F1
(All – 2190 queries)	0.573
(in KB – 1090 queries)	0.540
(not in KB – 1100 queries)	0.602
(NW docs – 1134 queries)	0.673
(WB docs – 343 queries)	0.494
(DF docs – 713 queries)	0.444
(PER – 686 queries)	0.537
(ORG – 701 queries)	0.619
(GPE – 803 queries)	0.562

Table 3: CASIA5 each score

From Table 3, we can see that our entity linking system achieves the comparable performance. The data set from discussion forum (DF) gets the lowest performance, the reason may be that the queries in DF are more dirty and misleading than other types of document, the recall of our candidate generation is only 0.81.

7 Conclusions

This paper describes our CASIA system in detail for KBP 2013 Entity linking task. Our system consist-

[1] http://www.nist.gov/tac/2013/KBP/EntityLinking/tools.html

s of four modules. 1) Query expansion module. 2) Candidate generation module. 3) Candidate Entity disambiguation module. 4) NIL clustering module. First, we expand the queries with the information of the query documents. Then we find the candidates of queries from the Knowledge Base using the Wikipedia knowledge and a string matching method. In the disambiguation module, we select a candidate entity using a supervised learning-to-rank method based on a variety of evidences. Finally, we cluster all the NIL entities which have not been detected in the above modules using the Hierarchical Agglomerative Clustering algorithm. The evaluation results show that our method is effective for English entity linking task. Finally, we use the NIL clustering module to cluster the NIL query. The evaluation of our five runs demonstrates the effectiveness of our system.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 61303180, No. 61070106, No. 61272332 and No. 61202329)

References

Mark Dredze, Paul McNamee, Delip Rao, Adam Gerber, and Tim Finin. 2010. Entity disambiguation for knowledge base population. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 277–285. Association for Computational Linguistics.

Xianpei Han and Le Sun. 2011. A generative entity-mention model for linking entities with knowledge base. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 945–954. Association for Computational Linguistics.

Heng Ji, Ralph Grishman, Hoa Trang Dang, Kira Griffitt, and Joe Ellis. 2010. Overview of the tac 2010 knowledge base population track. In *Third Text Analysis Conference (TAC 2010)*.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM.

David Milne and Ian H Witten. 2008. Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 509–518. ACM.

David Milne and Ian H Witten. 2012. An open-source toolkit for mining wikipedia. *Artificial Intelligence*.

Wei Shen, Jianyong Wang, Ping Luo, and Min Wang. 2012. Linden: linking named entities with knowledge base via semantic knowledge. In *Proceedings of the 21st international conference on World Wide Web*, pages 449–458. ACM.

I Witten and David Milne. 2008. An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In *Proceeding of AAAI Workshop on Wikipedia and Artificial Intelligence: an Evolving Synergy, AAAI Press, Chicago, USA*, pages 25–30.

Tao Zhang, Kang Liu, and Jun Zhao. 2012. The nlprir entity linking system at tac 2012.