Abstract

This document describes the work performed by the Universitat Politècnica de Catalunya (UPC) in its third participation at TAC-KBP 2014 in Mono-Lingual (English) Entity Linking task.

1 Introduction

EL is the task of referring a named entity mention occurring in a text document (henceforth, background document) to the unique entry within a reference knowledge base (KB). TAC-KBP track defines the task of EL as follows: having a set of queries, each one consisting of a query name along with a background document in which the query name can be found (the query name is mentioned using its begin and end offsets in the background document) and a source document collection from which systems can learn, the EL system is required to select the relevant KB entry. Queries sometimes consist of the same name string from different documents. The system is expected to distinguish the ambiguous names (e.g., Barcelona could refer to the sport team, the university, city, state, or person). An EL system is also required to cluster those named entities referring to a same Not-in-KB (NIL) entities. This Paper proposes a Vector Space Model (VSM) to rank a set of candidates which are possibly a correct answer for a query.

2 Literature Review

The recent works on EL in its contemporary history are inspired from the older history of Word Sense Disambiguation (WSD) where this challenge firstly arised. Many studies achieved on WSD are quite relevant to EL. Disambiguation methods in the state of the art can be classified into supervised methods, unsupervised methods and knowledge-based methods (Navigli, 1990).

Supervised Disambiguation (SD). The first category applies machine-learning techniques for inferring a classifier from training (manually annotated) data sets to classify new examples. Researcher proposed different methods for SD. A Decision List (Rivest, 1987) is a SD method containing a set of rules (if-then-else) to classify the samples. In continue, (Klivans and Servedio., 2006) used learning decision lists for Attribute Efficient Learning. (Magee, 1964) introduced another SD method Decision Tree that has a tree-like structure of decisions and their possible consequences. C4.5 (Quinlan, 1993), a common algorithm of learning decision trees was outperformed by other supervised methods (Mooney, 1996). (John and Langley, 1995) studied on the Naive Bayes classifier. This classifier is a supervised method based on the Bayes’ theorem and is a member of simple probabilistic classifiers. The model is based on the computing the conditional probability of each class membership depending on a set of features. (Mooney, 1996) demonstrated good performance of this classifier compared with other supervised methods. (McCulloch and Pitts, 1943) introduced Neural Networks that is a computational model inspired by central nervous system of organisms. The model is presented as a system of interconnected neurons. Although (Towell and Voorhees, 1998) showed an ap-
propriate performance by this model but the experiment was achieved in a small size of data. However, the dependency to large amount of training data is a major drawback (Navigli, 1990). Recently, different combination of supervised approaches are proposed. The combination methods are highly interesting since they can cover the weakness of each stand-alone SD methods (Navigli, 1990).

Unsupervised Disambiguation (UD). The underlying hypothesis of UD is that, each word is correlated with its neighboring context. Co-located words generate a cluster tending to a same sense or topic. No labeled training data set or any machine-readable resources (e.g. dictionary, ontology, thesauri, etc.) are applied for this approach (Navigli, 1990). Context Clustering (Schutze, 1992) is a UD method by which each occurrence of a target word in a corpus is indicated as a context vector. The vectors are then gathered in clusters, each indicating a sense of target word. A drawback of this method is that, a large amount of un-labeled training data is required. (Lin, 1998) studied on Word Clustering a UD method based on clustering the words which are semantically similar. Later on, (Pantel and Lin, 2002) proposed a word clustering approach called clustering by committee (CBC). (Widdows and Dorow, 2002) described another UD method Co-occurrence Graphs assuming that co-occurrence words and their relations generate a co-occurrence graph. In this graph, the vertices are co-occurrences and the edges are the relations between co-occurrences.

Knowledge-based Disambiguation (KD). The goal of this approach is to apply knowledge resources (such as dictionaries, thesauri, ontologies, collocations, etc.) for disambiguation (Lesk, 1986)(Banerjee and Pedersen, 2003)(Fu, 1982)(Bunke and Alberto Sanfeliu, 1990)(Mihalcea, 2004). Although, these methods have lower performance compared with supervised techniques, but they have a wider coverage (Navigli, 1990).

Recently, some collective efforts are done to research in this field in form of challenging competitions. The advantage of such competitions is that, the performance of systems are more comparable since all participants assess their systems in a same testbed including same resources and training and evaluation corpus. To this end, Knowledge Base Population (KBP) EL track at Text Analysis Conference (TAC) \(^1\) is the most important challenging competition being subject of significant study since 2009. The task is annually organized by which many teams present their proposed systems.

3 Methodology and Contribution

The method proposed in this paper follows the typical architecture in the state of the art (Figure 1). Briefly, given a query the system pre-processes the background document (Document Pre-processing step). Subsequently, those KB nodes which can be potential candidates to be the correct entity are selected (Candidate Generation step). Furthermore, a binary vector is generated from the contexts between both components of a set of pairs (query name, named entity mention) occurring in the sentence containing the begin and end offsets of query name in background document (henceforth, offset sentence) and also from all sentences of each candidate document–wikitext (Binary Vector Generation step). Next, a set of graphs is generated in order to represent relations between the components of the pairs (Graph Generation step). Finally, all candidates are ranked in a list and a candidate having the highest score is selected. In addition, all queries belonging to the same Not-In-KB (NIL) entity are clustered together assigning the same NIL id (Candidate Ranking and NIL clustering step). The final task (Candidate Ranking and NIL Clustering) is the most challenging and highly crucial among steps above.

Details of each step are provided next.

3.1 Document Pre-processing

Initially, the background document has to be normalized to be used by other components. To this objective, the system pre-processes the document in the following way.

Document Partitioning and Text Cleaning. This component separates the textual and non-textual parts of the document. Then, the further steps are only applied over the textual part.

In addition, each document might contain HTML tags and noise (e.g. in Web documents) which are

\(^1\)The TAC is organized and sponsored by the U.S. National Institute of Standards and Technology (NIST) and the U.S. Department of Defense.
removed by the system.

**Sentence Segmentation and Text Normalization.** This module operates on the context of documents as following:

- *Sentence Segmentation.* The documents are splitted by discovering sentence boundaries.
- *Soft Mention (SM).* Entity mentions represented with abbreviations are expanded, e.g. “Tech. Univ. of Texas”, is replaced with “Technical University of Texas”. To this end, a dictionary-based mapping was manually generated and applied.

### 3.2 Query Expansion

In most queries, query name might be ambiguous, or background document contains poor and sparse information about the query. In these cases, query expansion can reduce the ambiguity of query name and enrich the content of documents through finding name variants of the query name, integrating more discriminative information, and tagging meta data to the content of documents.

For doing so, we apply the following steps:

**Query Classification.** Query type recognition helps to filter out those KB entities with type different to the query type. Our system classifies queries into 3 entity types: PER (e.g. “George Washington”), ORG (e.g. “Microsoft”) and GPE (Geopolitical Entity, e.g. “Heidelberg city”). We proceed under the assumption that a longer mention (e.g. “George Washington”) tends to be less ambiguous than a shorter one (e.g. “Washington”) and, thus, the type of the longest query mention tends to be the correct one. The query classification is performed in three steps. First, we use the Illinois Named Entity Recognizer and Classifier (NERC) (Ratinov and Roth, 2009) to tag the types of named entity mentions occurring in the background document. Second, we find the set of mentions in the background document referring to the query. More concretely, we take mention $m_1$ defined by the query offsets within the background document (e.g. “Bush”) and find the set of mentions that include $m_1$ occurring in that document (e.g. “Bush”, “G. Bush”, “George W. Bush”). Finally, we select the longest mention from the resulting set of mentions and take its type as the query type.

**Alternate Name Generation.** Generating Alternate Names (AN) of each query can effectively reduce the ambiguities of the mention and improve recall, under the assumption that two name variants in the same document can refer to the same entity. We follow the techniques below to generate AN:

- *Acronym Expansion.* Acronyms form a major part of ORG queries and can be highly ambiguous, e.g. “ABC” is referred to around 100 entities. The purpose of acronym expansion is to reduce its ambiguity. The system seeks inside the background document to gather all subsequent tokens with the first capital orderly matched to the letters of the acronym. Also, the expansions are acquired before or inside the parentheses, e.g. “Congolese National Police (PNC)”, or “PNC (Congolese National Police)”.  
- *Gazetteer-based AN Generation.* Sometimes, query names are abbreviations. In these occasions, auxiliary gazetteers are beneficial to map the pairs of $\langle$abbreviation, expansion$\rangle$ such as the US states, (e.g. the pair $\langle$CA, California$\rangle$ or $\langle$MD, Maryland$\rangle$), and country abbreviations, (e.g. the pairs $\langle$UK, United Kingdom$\rangle$, $\langle$US, United States$\rangle$ or $\langle$UAE, United Arab Emirates$\rangle$).
Google API. In more challenging cases, some query names contain grammatical irregularities or a partial form of the entity name. Using Google API, more complete forms of the query name are probed across the Web. For doing so, the component captures title of first (top ranked) result of the Google search engine as possibly better form of the query name. For instance, in the case of query name “Man U”, using the method above, the complete form “Manchester United F.C.” is obtained.

3.3 Candidate Generation

Given a particular query, \( q \), a set of candidates, \( C = \{c_1, \ldots, c_n\} \), is found by retrieving those entities from the KB whose names are similar enough, using Dice measure, to one of the alternate names of \( q \) found by the query expansion. In our experiments we used a similarity threshold of 0.9, 0.8 and 1 for PER, ORG and GPE respectively. By comparing the candidate entity type extracted from the corresponding KB page and query type obtained by NERC, we filter out those candidates having different types to attain more discriminative candidates.

3.4 Binary Vector Generation

In this step, we exploit the context between components of the pair \( \langle \text{query name}, \text{named entity mention} \rangle \) occurring in the offset sentence of background document and all sentences of each candidate wikitext. Our hypothesis is based on the fact that the context between the components of each pair contains discriminative information and can be used to rank candidates. In the case of background document the system just used the offset sentence instead of all sentences since in most cases the named entity mentions occurring in other sentences of background document are far from the main topic and may have a negative influence in ranking candidates.

Consider a query name \( q \) along with its background document \( d_q \) in which the query name occurs, the offset sentence \( s_o \) and a set of named entity mentions occurring in the offset sentence \( M_{s_o} = \{m_q^1, \ldots, m_q^\ell\} \). The system extracts each pair \( \lambda_i \) composed by the query name \( q \) and each named entity mention \( m_q^i \in M_{s_o} \):

\[
\lambda_i = \langle q, m_q^i \rangle \quad (1)
\]

where, \( i \in \{1, \ldots, |M_{s_o}|\} \).

Consider the set of candidates \( C = \{c_1, \ldots, c_n\} \). Let the set of all sentences in each candidate’s wikitext \( S_c = \{s^1_c, \ldots, s^\ell_c\} \), and a set of named entity mentions in each candidate’s wikitext \( M_c = \{m^1_c, \ldots, m^\ell_c\} \), the system extracts each pair \( \lambda \) consists of query name and a mention both occurring in the same sentence:

\[
\lambda_{i,j,k} = \langle q_{i,j}, m_{i,j,k} \rangle \quad (2)
\]

where, \( i \in \{1, \ldots, |C|\}, j \in \{1, \ldots, |S_c|\}, \) and \( k \in \{1, \ldots, |M_j|\} \), i.e. \( m_{i,j,k} \) is the \( k \)-th mention in \( j \)-th sentence of \( i \)-th candidate. Assuming the context between occurrences \( q \) and \( m \) defined as follows:

\[
W_\lambda = w_1w_2\ldots w_n \quad (3)
\]

The word sequence \( W_\lambda \) is then lemmatized, and all stopwords are removed. The system gathers all lemmas of all pairs together to create a bag of lemmas.

\[
L_\lambda = \{l_1, l_2, \ldots, l_y\} \quad (4)
\]

\[
L_T = \bigcup_{\lambda \in \Lambda} L_\lambda \quad (5)
\]

where \( L_\lambda \) is a bag of lemmas of each pair \( \lambda \), \( \Lambda \) is a set of all existing pairs, \( L_T \) is a bag of lemmas of all pairs \( \Lambda \). Next, the system generates for each pair a vector of features using the bag of lemmas (binary vectors). For doing so, the system generates a binary vector (a row matrix) \( \varphi_i \) assigned to pair \( \lambda_i \) (Equation 6). The value of each element of the vectors is initially set to zero. The number of vectors is equal to the number of pairs (\(|\Lambda|\)) and the number of elements of each vector is equal to the number of lemmas in \( L_T \) (\(|L_T|\)). For each vector, if the system finds a lemma from the bag of lemmas (\( L_T \)) in the relevant \( L_\lambda \), the corresponding element of that vector is set to one (Equation 6).

\[
\varphi_i = \begin{bmatrix} l_1 & l_2 & \ldots & l_d \end{bmatrix} \quad (6)
\]

Each element in vectors is equal to:

\[
b_i^j = \begin{cases} 0 & \text{if } l_j \text{ not in } L_\lambda_i \\ 1 & \text{if } l_j \text{ in } L_\lambda_i \end{cases} \quad (7)
\]
G to measure the similarity between ure 2). This step aims to generate a graph structure are the sets of vertices and edges respectively (Fig-

3.5 Graph Generation

In order to rank the candidates, the system generates a star graph for the query \( G_q = (V_q, E_q) \) and for each candidate \( G_c = (V_c, E_c) \) in which \( V \) and \( E \) are the sets of vertices and edges respectively (Figure 2). This step aims to generate a graph structure to measure the similarity between \( G_q \) and each \( G_c \) in order to select the most similar candidate to the query. Central vertex of \( G_q \) is labeled with the query name and central vertex of each \( G_c \) is labeled with the candidate name. Other vertices in \( G_q \) and \( G_c \) are labeled with those named entity mentions existing in the set of pairs \( \lambda \in \Lambda \). Each edge is labeled with the semantic relation existing between the linked entities which is represented by a binary vector \( \varphi \) corresponding to each pair \( \lambda \). Given that we are dealing with star graphs, each graph \( G_i \) with central vertex \( i \) can be represented as the list of pairs \( (e_j, v_j) \) for all outcoming edges of vertex \( i \) (Equation 8).

\[
G_i := \{(e_j, v_j)\}_i = \{(\varphi_j, m_j)\}_i \quad (8)
\]

3.6 Candidate Ranking and NIL Clustering

For ranking candidates, each \( G_{c \in C} \) is scored based on the similarity between \( G_q \) and \( G_c \), which is equal to the degree of similarity between outcoming edges of both graphs \( G_q \) and \( G_c \) (Equation 9).

\[
Sim(G_q, G_c) = Sim(\{(\varphi_i, m_i)\}_q, \{(\varphi_j, m_j)\}_c) \quad (9)
\]

In order to calculate \( Sim(G_q, G_c) \), the system first compares the similarity \( \beta \) between each vertex \( m_i \) \( G_q \) and each vertex \( m_j \in G_c \) (except \( q \) and \( c_i \)) using Levenshtein distance metric (Equation 10).

\[
\beta_{m_i, m_j} = 1 - \frac{lev_{m_i, m_j}}{|m_i| + |m_j|} \quad (10)
\]

where \( \beta_{m_i, m_j} \) is the degree of similarity between \( m_i \in G_q \) and \( m_j \in G_c \), and \( lev_{m_i, m_j} \) is Levenshtein metric for measuring the difference between two strings \( m_i \) and \( m_j \), and \( |m_i| \) and \( |m_j| \) are lengths (number of characters) of \( m_i \) and \( m_j \) respectively. For instance, if \( m_i = \text{"Barcelona"} \) and \( m_j = \text{"F.C. Barcelona"} \), then, \( lev_{m_i, m_j} = 5 \) and \( |m_i| + |m_j| = 23 \), therefore, \( \beta_{m_i, m_j} = 1 - \frac{5}{23} = 0.78 \).

In addition, the system compares the similarity \( \beta \) between each edge \( \varphi_i \in G_q \) and each edge \( \varphi_j \in G_c \) using Dice metric (Equation 11).

\[
\beta_{\varphi_i, \varphi_j} = dice_{\varphi_i, \varphi_j} = \frac{2T_{i,j}}{T_i + T_j} \quad (11)
\]

where \( \beta_{\varphi_i, \varphi_j} \) is the degree of similarity between \( \varphi_i \in G_q \) and \( \varphi_j \in G_c \), and \( dice_{\varphi_i, \varphi_j} \) is the function to calculate Dice coefficient between \( \varphi_i \) and \( \varphi_j \), \( T_{i,j} \) is the number of positive matches between vectors \( \varphi_i \) and \( \varphi_j \), and \( T_i \) and \( T_j \) are the total number of positive presences in the vectors \( \varphi_i \) and \( \varphi_j \) respectively. For instance in Equation 11, suppose that \( \varphi_i = [1110001010] \) and \( \varphi_j = [0010001011] \), then, \( dice_{\varphi_i, \varphi_j} = \frac{2(3)}{9} = 0.66 \).

Furthermore, for each \( G_c \) the system generates a set of links \( H_{q,c} = \{h_1, \ldots, h_f\} \), each link \( h \) between vertices \( m_q \) and \( m_c \). As shown in Figure 3 each link \( h \) has attached weight \( \alpha \). To calculate the value of each \( \alpha \), the system combines the similarities \( \beta_{m_i, m_j} \) and \( \beta_{\varphi_i, \varphi_j} \) (Equation 12).

\[
\alpha_{h \in H_{q,c}} = \beta_{m_i, m_j} + (1 - \beta_{m_i, m_j})\beta_{\varphi_i, \varphi_j} \quad (12)
\]

Subsequently, to score each candidate, the average of all \( \alpha \) values for each \( G_c \) is obtained:

\[
Sim(G_q, G_c) = X_{c \in C} = \frac{\sum_{h \in H} \alpha_h}{|H|} \quad (13)
\]

where \( X_{c \in C} \) indicates the score obtained by the candidate \( c \in C \). The system then selects that candidate

---

Footnote: In this example we assumed \(|\varphi| = 10\), however in the real samples, \(|\varphi| \) is much more than this amount (usually, \(|\varphi| > 100\)).
having the highest score as the correct reference of the query (Equation 14).

\[
\text{answer}_q := \{ z \in C | \forall c \in C : X_c \leq X_z \}
\] (14)

where \(\text{answer}_q\) indicates those entry in the KB related to the query.

For those queries referring to entities which are not in the KB (NIL queries), the system should cluster them in groups, each referring to a same Not-In-KB entity (NIL Clustering). To this objective, a term clustering method is applied to cluster such queries. Each initial NIL query forms a cluster assigning a NIL id. The module compares each new NIL query with each existing cluster (initial NIL query) using a dice coefficient similarity between all ANs (including query name) of both queries. In the comparison between new NIL query and the cluster, we consider the maximum dice score. If the similarity is higher than the predefined NIL threshold, the new NIL query obtains the Id of this cluster, otherwise it forms a new NIL cluster obtaining a new NIL id. In our experiments we manually selected 0.8 as NIL threshold (Equation 15).

\[
id_{\text{nil}} = \begin{cases} 
id_{\text{clr}} & \text{if } \text{dice}_{\text{nil,clr}} \geq 0.8 \\
id_{\text{nclr}} & \text{otherwise} \end{cases}
\] (15)

where, \(id_{\text{nil}}\) is the Id of NIL query, \(id_{\text{clr}}\) is the Id of an existing cluster, \(\text{dice}_{\text{nil,clr}}\) is Dice function applied to NIL query and existing cluster, and \(id_{\text{nclr}}\) is Id of a new cluster.

4 Evaluation Framework

We have participated in the framework of the TAC-KBP 2014 Mono-Lingual (English) EL evaluation tracks.

Given a list of queries, the system should provide the identifier of the KB entry to which the name refers if existing, or a NIL ID if there is no such KB entry. The EL system is also required to cluster together queries referring to the same non-KB (NIL) entities and provide a unique ID for each cluster. The reference KB used in this track includes hundreds of thousands of entities based on articles from an October 2008 dump of English WP, which includes 818,741 nodes. The evaluation query set in 2014 experiment contains 5234 queries. Some entities share confusable names, especially challenging in the case of acronyms.

5 Results and Analysis

We have participated in TAC-KBP 2014 Mono-Lingual (English) EL track. Totally, 17 teams submitted 55 runs in this section. Table 1 illustrates our official results measured by B-cubed+ metric (F1) and submitted in two runs. First run without access to the Web and the second run using web access. The table splits the results by those query answers existing in reference KB (in-KB) and those not in the KB (NIL). Evaluation corpus in TAC-KBP 2014 includes three types of genres, News Wires (NW), Web Documents (WB), and Discussion Fora (DF). DF is highly challenging since it contains many grammatical irregularities extracted from fora, blogs, etc. Furthermore, the table indicates the results by three different query types, PER, ORG, and GPE. It also shows results obtained by the median of all participants and results of a team having highest score in TAC-KBP 2014.

Although, our assumption was to obviously obtain a better results in run 2 (with access to web), but the overall results obtained from two runs are very close. By analogy, the system obtained a higher score to link in-KB queries and a lower score for NIL queries in run 2 compared against other run. Our results are less than the median of all participants in this track. Our result for NIL queries are

\[http://www.nist.gov/tac/\]
Table 1: TALP official results obtained in TAK-KBP 2014 Mono-Lingual (English) EL evaluation track submitted in two runs. Run 1 without web access and run 2 with web access.

near to the median but the result for in-KB queries is far from the score obtained by the median. In addition, among the query types the worst result belongs to the GPE.

6 Conclusions and Future Work

In our third participation in TAC-KBP 2014 EL evaluation track, we applied a VSM model in order to rank the candidates of each query. The results were compared with those obtained by all participants. We achieved the comparison using B-cubed+F1 metric. We obtained a result less that the median of all participants in TAC-KBP 2014. In order to improve the system accuracy, we aim to combine another graph-based ranker. This ranker will measure the semantic similarity between the query and each candidate. It would be beneficial when the queries are highly ambiguous.

Acknowledgements

This work has been produced with the support of the SKATER project (TIN2012-38584-C06-01).

References


John F. Magee. 1964. *Decision trees for decision making*. Graduate School of Business Administration, Harvard University.


Patrick Pantel and Dekang Lin. 2002. Discovering word senses from text. In *the 8th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM.


