The TAI System for Trilingual Entity Discovery and Linking Track in TAC KBP 2017

Tao Yang, Dong Du and Feng zhang
Tencent AI Platform Department,
100190, Beijing, P.R. China
{rigorosyang, dongdu, jayzhang}@tencent.com

Abstract

In this paper we propose the TAI's system for the Tri-lingual Entity Discovery and Linking (EDL) track in TAC-KBP 2017. EDL track in TAC KBP 2017 is to extract named and nominal entity mentions from a collection of textual documents in three languages (Chinese, English and Spanish), and link them to an existing Knowledge Base (BaseKB). To tackle this problem, we propose a neural-network-based system, which is called TAI's system here. TAI's system consists of two parts: Mention Detection sub-system (MD) and Entity Linking sub-system (EL). MD contains a two-layers stacked deep bi-directional Long Short Term Memory (BiLSTM) to encode the input information and a Conditional Random Fields (CRF) to produce the entity tags. Besides, we also consider many useful features into this framework to improve the performance of ED. EL consists of five modules: candidates generation module, pair-wise ranking module, NOM Resolution module, NIL prediction module and NIL cluster module. We conduct experiments on the released data sets. The experimental results show that TAI's system achieves 70.5 of F1 score in the typed_mention_ceaf and the 67.4 of F1 score in the typed_mention_ceaf_plus.

1 Introduction

The task of EDL track in TAC KBP 2017 is required to automatically identify and classify named entity mentions into five pre-defined entity from a collection of textual documents in three kinds of languages (Chinese, English and Spanish), and then link each mention to an existing Knowledge Base (BaseKB). An EDL system is also required to cluster the NIL mentions. The predefined entity types are: Person (PER), Geo-political Entity (GPE), Organization (ORG), Facility (FAC) and Location (LOC), and the mention types are Named (NAM) and Nominal (NOM). Besides, the textual documents come from Newswire (NW) and Discussion Forum (DF) and 500 core documents are manually annotated for evaluation.

2 Overview of TAI's System

The architecture of TAI's system is shown in Figure 1. TAI's system consists of two parts: Mention Detection sub-system (MD) and Entity Linking sub-system (EL). The MD contains preprocessing, mention extraction. The EL consists of candidates generation, candidates ranking, NIL prediction, NOM resolution and NIL cluster.

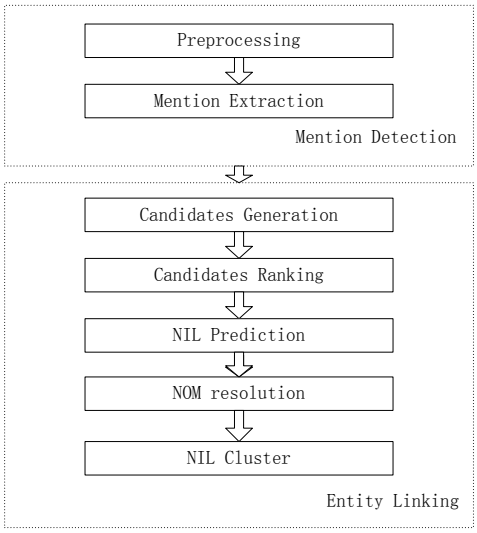

Figure 1: The architecture of TAI's system

3 Mention Detection

3.1 Preprocessing

The training data used here is the training and evaluation golden data used in 2015 and 2016 EDL track. In preprocess step, we firstly delete the xml tags in the raw texts and the "<quote >" regions from the forum texts. For Chinese, traditional characters are converted to simplified characters. Then the cleaned texts are tokenized by using the corenlp tool (Manning et al., 2014). Author elements are also extracted. Furthermore, in order to reduce the error of Chinese word segment, we use the character sequence instead of the word sequence in Chinese language.

3.2 The Architecture of Mention Detection

We treat mention detection as a sequence labeling problem which each word aligned with one tag. There is a strong dependency between the adjacent tags, and conditional random fields (CRF) (Lafferty et al., 2001) are widely used to settle this problem. However, the disadvantage of CRF is that it needs complicated feature engineering work , which is very time-consuming. In order to reduce the manual work in feature extraction, recently, deep neural networks have been applied to obtain effective features of sentences. Therefore in this work we use the classical neural BiLSTM + CRF neural network model to produce the label sequence, which have been successfully applied to Named Entity Recognition (Lample et al., 2016). To further improve the performance, we adopt a two-layers stacked deep BiLSTM. The architecture of mention detection is shown in Figure 2. NOM and NAM mentions are treated as the same and jointly detected altogether in a single model, except for that we extract specific features for NOM mentions.

There are three layers in our model: input layer, representation layer and output layer.

Input Layer. We denote $X = \{x_1, x_2, ...w_n\}$ and $Y = \{y_1, y_2, ...y_n\}$ as the input and output sequence. This layer aims to generate the input feature by concatenating different raw features. There are three kinds of raw features: the word embedding, the character embedding and the additional features. Section 3.3 will elaborate these features. These features are concatenated to generate the final input feature as follows:

$$V_i = [V_{i,w}, V_{i,c}, V_{i,a}, ...] \tag{1}$$

Representation Layer. This layer aims to generate the high level abstraction representation, which is implemented by a two-layers stacked BiLSTM:

$$\overrightarrow{h_i} = two\_layer\_LSTM(V_i) \tag{2}$$

$$\overleftarrow{h_i} = two\_layer\_LSTM(V_i) \tag{3}$$

$$h_i = [\overrightarrow{h_i}, \overleftarrow{h_i}] \tag{4}$$

Besides, we add the skip connection between the input layer and the second BiLSTM layer to alleviate the gradient vanishing problem (Wu et al., 2016).

Output Layer. Considering the correlation between adjacent tag, we use a linear-chain CRF model as our output layer:

$$O_i = W h_i \tag{5}$$

$$s(X, Y) = \sum_{i=1}^{n} O_{i,y_i} + \sum_{i=0}^{n-1} T_{y_i, y_{i+1}} \tag{6}$$

$$p(Y|X) = \frac{exp(s(X, Y))}{\sum\limits_{y \in all\_y} exp(s(X, y))} \tag{7}$$

$$L = -log(p(Y|X)) \tag{8}$$

where W is a $k \times 2p$ matrix, k is this number of distinct output tags and p is the second LSTM's hidden unit number. O is the $n \times k$ observation matrix, T is the transition parameter matrix which is learned during training. Since we only model the bigram dependency of output tags, Forward and Viterbi algorithm can be used for training and inference. The tag scheme we used here is the BIEOS scheme.

3.3 Features

This model has millions of parameters and the training data is quite small. So only using the end-to-end model with word embedding is not enough for the task. Therefor, we consider following features into the model.

1. **Word Embedding**. We used the pre-trained word embedding instead of random initialization word embedding. It is trained by using the wang2vec tool (Ling et al., 2015), which has a better performance than word2vec (Mikolov et al., 2013). We also found that pre-training on Gigaword instead

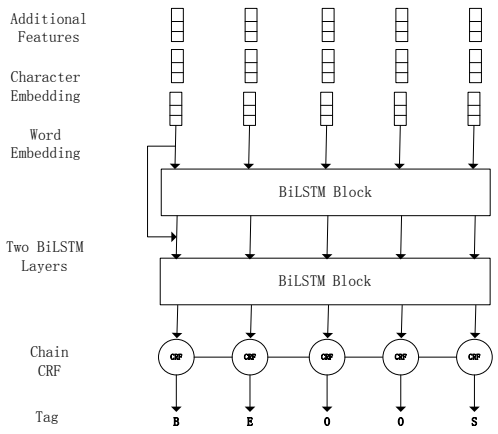

Figure 2: The architecture of Mention Extraction

of Wikipedia yields a boost on the performance. The reason is that our newsware prediction texts are quite similar to Gigaword source. Beside the character embedding for Chinese, we still generate the word embedding feature.

2. **Character Embedding**. For English and Spanish words, character representation is necessary, which could be used to reduce Out Of Vocabulary(OOV) problem. We use another BiLSTM to encode the character embedding (Lample et al., 2016). The architecture is shown in Figure 3. Since Chinese character is quite ambiguity, we use the positional character embedding to alleviate this problem (Chen et al., 2015), which considers the character's position information during training the character and word embedding.

3. **Dictionary Feature**. We collected entities form Wikipedia and Baike as the dictionary feature. The entities include person,location, organization and facility types.

4. **POS and NER Feature**. For English and Spanish, we use corenlp (Manning et al., 2014) to generate the POS and NER results, which are used as kinds of features. For Chinese, we use our company's $qqseg$ to produce the POS and NER features.
$qqseg$ can segment Chinese texts and generate the POS and NER results, and the NER type includes PER, ORG, LOC and other types. It is trained using the public and private resources. The public resource consists of China Daily, Microsoft's label data and so on; the private resource consists of the internal business and label data, which is quite large. The NER model is CRF model, and the segmenter's algorithm is word grid search plus language model to decide the best path.

5. **Word Boundary Feature**. For Chinese, we add the word boundary feature to indicates whether this character is at the word's boundary or in the word.

6. **Nom's Feature**. We found some signals are very high for the NOM detection. Such as "a" , "The", "一些","几个". To better recognize the NOM entity, we create such features for some selected popular NOMs in the 2016 evaluation data, such as "president" and "国家".

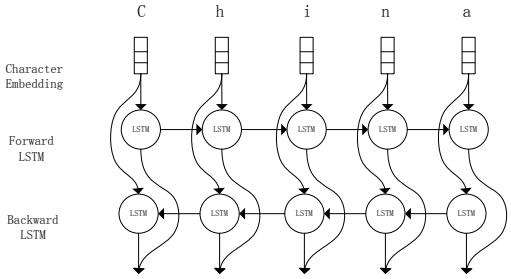

Figure 3: The architecture of character embedding

3.4 Model Configurations

All the parameters are selected based on the 5-fold cross validation. We set batch size to 8, epoches to 50, initial learning rate to 0.01. Learning rate will decay linear proportional to the running epoch number. For Chinese, English and Spanish, we set the word embedding dimensions to 100,100,64 and the LSTM hidden unit number to 200, 100 and 128. The character embedding are randomly initialized, and dimensions for English and Spanish are set to 25 and 32. All the additional feature's dimension are set to 10. Dropout rate for the input layer is set to 0.5, all numbers are translated to zero before training.
To train the UNK word, we randomly make the singletons in the training data to be UNK as (Lample et al., 2016) did. Parameter optimization is performed using the SGD, and early stop is used to prevent the over-fitting.
Moreover we also tried to use the ensemble learning to further improve the performance of mention detection. for each language we randomly select

four parts of the whole training data from the five-fold splits twice to train two models with different parameters initialization.

4 Entity Linking

In the entity linking (EL) task, each detected mention (including NAM and NOM) should be linked to an entity in the provided knowledge base (namely the BaseKB), or labeled as NIL if it cannot be found in the knowledge base. Finally all the NILs should be clustered. Our EL system has five sub modules.

4.1 Candidates Generation

In this module we need to generate candidate entities for each detected mention. This module is crucial for the EL performance because it greatly influence the recall.

In the first step we offline built the alias-to-entities dictionary. Specifically, firstly the mapping between the entity's name and id in BaseKB are added into this dictionary. Then we need to mine as many as possible aliases for these entities. We mine those entities' aliases from the Wikipedia's title, anchor texts, disambiguate pages and redirect information. We also add some aliases from Baike and Google translation service for Chinese. Besides, we also split the PER entity's name into substrings to generate its aliases.

For each NAM mention, The way to generate its candidate entities set is as follows: firstly we match the mention string with the aliases in this dictionary, and all the candidate entities are added into the set if found. Then Fuzzy matching is enabled to search again. We also search the whole documents to find whether it is other mentions' substring, such as Bush and George Bush. If it is found, all the longer mention's candidate entities will be added into the set.

If the candidate entities set is empty, make it NIL directly. If the candidate entity set size is more than 100, it needs to be truncated. The truncation is based on the entity's prior popularity feature.

4.2 Candidates Ranking

For each NAM mention, we need to identify its target entity from the candidate entities which generated in section 4.1. We adopt the pairwise rank model lambdaMART to settle the problem. Specifically, the ranking pair consists of the target entity e_t and the non-target entity e_i: $< e_t, e_i >$, e_t should be ranking higher than all the e_i. Many handcrafted features are created, including the mention level feature, entity level feature, mention-to-entity feature and entity-to-entity feature. The features are listed as follows:

1. **PageRank-Based Popular**. This feature can be used to measure the prior importance of the given entity. It is calculated by page rank algorithm based on the link information of Wikipedia's anchors and the structural information of BaseKB.

2. **Language Number**. The feature is the language numbers of this Wikipedia's page. It is still a popular class feature, there are more language numbers, it's more probably popular.

3. **Mention Link Probability**. It is calculated by the Wikipedia's anchors as follows:

$$link\_prob(m, c) = \frac{count(m, c)}{\sum_{c'} count(m, c')} \quad (9)$$

   m is the mention and the c is the linked page(entity). $count(m, c)$ is the total co-occurrence count of m and c.

4. **Document Type**. The document type , NW or DF.

5. **Mention's Entity Type**. The entity type which are extracted by mention detection, including PER, LOC, ORG, FAC and GPE.

6. **BaseKB's Entity Type**. Some selected BaseKB entity types, as showed in table 1.

7. **Entity's Properties appeared in the Context**. This feature indicates that whether the properties for some entity appearing in it's context.

8. **Word similarity between the Entity and the Context based on Bow**. Entity's description and the entity's context are represented by the bag of words, and then we compute the jaccard and cosine similarity score between the entity's description and the entity's context.

9. **Semantic similarity between the Entity and the Context based on DSSM model**.

We use DSSM model (Deep Structured Semantic Model)(Huang et al., 2013) to produce entities' semantic representation. Firstly, we use the Wikipedia's text to represent each entity, the anchor as the target entity, the surrounding words as the context. Then DSSM model is pre-trained using these data. After that the network is fine-tuned by the KBP training data. When predicting, we will calculate the semantic similarity score between each candidate entity and its context. The DSSM architecture is shown in Figure 4. pair-wise ranking loss is adopted:

$$L = max\{0, M - (cos(e_t, c) - (cos(e_i, c)))\} \quad (10)$$

10. **Max WLM Score**. The max WLM score(Milne, 2007) between current entity and the other mentions' candidate entities. the WLM score is calculated as follows:

$$WLM(e_1, e_2) = 1 - \frac{log(max(|S(e_1)|, |S(e_2)|) - max(|S(e_1) \bigcap S(e_2)|))}{log(|W|) - log(min(|S(e_1)|, |S(e_2)|))} \quad (11)$$

The W means the whole page set, the $S(e_1)$ is the page set linked to e_1. This score is widely used to represent the semantic relatedness score between entities.

11. **Global Coherent Score**. The features mentioned above are local level features, which consider entities independently (the WLM feature considered the entity-to-entity semantic relatedness, but it is still entity independent). Semantic associations exists between entities in a document, so collective disambiguation methods are proposed to jointly disambiguate entities in a document. Solving this problem is a NP-hard problem and the classic method is the graph-based model proposed in (Han et al., 2011).
There are two types of edges in the graph: mention-to-entity edge and entity-to-entity edge. The weight of mention-to-entity edge is the word similarity between the entity and the context based on bow, the weight of entity-to-entity edge is the WLM score. All the weights are normalized as follows:

$$P(m \rightarrow e) = \frac{Sim(m, e)}{\sum_{e \in N_m} Sim(m, e)} \quad (12)$$

$$P(e_i \rightarrow e_j) = \frac{WLM(e_i, e_j)}{\sum_{e \in N_{e_i}} WLM(e_i, e)} \quad (13)$$

where N_m is the set of the neighbor entities of mention m, N_e is the set of neighbor entities of entity e. After the graph is created, personalized page rank algorithm is adopted as described in (Han et al., 2011).

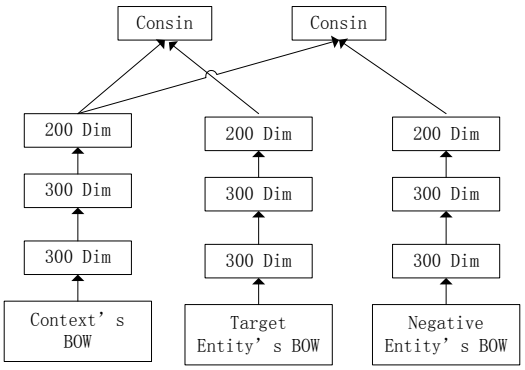

Figure 4: The architecture of DSSM

We tried two settings for entity linking, one is to learn the ranking model in separate languages, and the other is to combine the three languages' training data to learn.

4.3 NIL Prediction

The target entity for this mention may be not in the candidate entity set, namely the first ranked entity is not the right answer. So we need to validation it.
We adopt another binary classification model to make the decision. Apart from the features used by the ranking model, additional features are added. They are the first ranked entity's score, the differential between the top score and the second place's score, the standard error of all the scores and so on.

4.4 NOM Resolution

The ranking and NIL prediction models mentioned above are focusing on resolving the NAM mentions' target entities. We also need to resolve the NOM mentions'. Many rules are proposed to decide the NOMs link id. For Instance we can directly link "government" in the "the government of United States" from the pre-compiled dictionary; link the NOM to the nearest NAM with the

organization.organization
location.location
geography
location.country
location.administrative
division
location.statistical_region
people.person
architecture.structure
government.governmental_body
base.newsevents.news_reporting_organisation
government.government
government.legislative_committee
aviation.airport
education.educational_institution
base.prison.prison
government.governmental_jurisdiction

Table 1: The selected entity type in BaseKB as EL ranking features.

same type such as " 主席" in the " 国家主席习近平"; select the most frequent GPE for the GPE NOM in this document and so on.
Expect these rules, we also train a simple binary classification model. We collected all the NAMs which appear in the NOM's context and then create features for each NOM and NAM pair, to classify whether this NOM linked to this NAM. This model's feature includes the word bigrams, POS and the dependency paths.
If the NOM is still not linked, make it NIL.

4.5 NIL Cluster

In the final step we need to cluster all the NILs. This is mainly based on rules. Specifically for all the NIL mention in the same doc, we cluster them according to the mention string. Authors and the text's NIL mentions are clustered altogether. For person mention, if mention string partially match, we also cluster them. Some special rules are also created. For example, for the "楼主" in CMN DF, we always cluster it with the first author.
In this module we also correct the entity type based on rules. Specifically, we build the link id to entity type dictionary based on the 2015 and 2016 golden data which has high confidence, such as EU to GPE. We adopt this dictionary to correct the entity type.

5 Results

Because the number of model's parameter is much larger than the training data, we select 30 percent training data as the development data to prevent over-fitting (has_dev). The disadvantage of this method is that it cannot train the model with the whole training data set. To settle the problem, we also train the model by using the entire training data with dropout mechanics to prevent the overfitting (no_dev). For Spanish's mention detection, one setting is that combining the English and Spanish training data (spa_combine).
As for entity linking task, we has two settings: one is to learn the ranking model in separate languages (el_separate), and the other is to combine the three languages' training data to learn (el_combined).
Three kinds of combinations are experimented: no_dev + el_separate, has_dev + el_combined, has_dev + el_combined + spa_combine. The third one achieves 69.7 F1 score which is the best among the three.

5.1 Predicted NIL NOM's impact on the mention_ceaf

The five-fold cross validation experimental results on the 2016's evaluation data shows that if we remove the NIL NOM from the final results, the mention_ceaf score will improve about 1 point stably. One possible reason is that removing NIL NOM will reduce the strong typed mention match F1 score, but the precision is better, and the mention_ceaf class metric is quite sensitive to the mention detection's precision or the NOM's precision. To confirm this, we generate a new run by removing all the NIL NOM from the above option two. The final results agree with our conclusion, although the mention detection's performance degrades, the typed_mention_ceaf metric grows up. the new run is our best run according to the type_mention_ceaf metric, even better than the above option three. table 2 shows the detailed results.

6 Conclusion

In EDL track of TAC BKP 2017, we propose TAI's system to settle the problem, which contains Entity Mention Detection sub-system (EMD) and Entity Linking sub-system (EL). The experimental results show the effectiveness of our proposed method. But it still has shortcoming. In the future work, we can explore some more useful mod-

		strong_typed_mention_ceaf			strong_typed_all_match			typed_mention_ceaf		
Language	Run	*Prec.*	*Rec.*	*F1*	*Prec.*	*Rec.*	*F1*	*Prec.*	*Rec.*	*F1*
Trilingual	option two	0.813	0.753	0.782	0.700	0.648	0.673	0.719	0.666	0.691
Trilingual	new run	0.850	0.686	0.759	0.760	0.613	0.678	0.790	0.637	0.705
CMN	option two	0.839	0.721	0.775	0.763	0.655	0.705	0.782	0.672	0.723
CMN	new run	0.871	0.695	0.773	0.800	0.638	0.710	0.824	0.658	0.732
ENG	option two	0.779	0.789	0.784	0.664	0.672	0.668	0.683	0.692	0.688
ENG	new run	0.817	0.692	0.749	0.721	0.611	0.662	0.761	0.645	0.699
SPA	option two	0.815	0.764	0.789	0.655	0.615	0.634	0.692	0.649	0.670
SPA	new run	0.855	0.667	0.749	0.742	0.578	0.650	0.787	0.613	0.689

Table 2: The comparison between the option two and the new run.

el to improve the NOM resolution. Besides, NOM mentions' detection also needs to be further improved.

References

Xinxiong Chen, Lei Xu, Zhiyuan Liu, Maosong Sun, and Huan-Bo Luan. 2015. Joint learning of character and word embeddings. In *IJCAI*. pages 1236–1242.

Xianpei Han, Le Sun, and Jun Zhao. 2011. Collective entity linking in web text: a graph-based method. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. ACM, pages 765–774.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*. ACM, pages 2333–2338.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data .

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360* .

Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 1299–1304.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*. pages 55–60.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

David Milne. 2007. Computing semantic relatedness using wikipedia link structure. In *Proceedings of the new zealand computer science research student conference*.

Huijia Wu, Jiajun Zhang, and Chengqing Zong. 2016. An empirical exploration of skip connections for sequential tagging. *arXiv preprint arXiv:1610.03167* .