The YorkNRM Systems for Trilingual EDL Tasks at TAC KBP 2016

Mingbin Xu, Feng Wei, Sedtawut Watcharawittayakul, Yuchen Kang, Hui Jiang
Department of Electrical Engineering and Computer Science
Lassonde School of Engineering, York University
4700 Keele Street, Toronto, Ontario, Canada
{xmb,fwei,watchara,victorka,hj}@cse.yorku.ca

Abstract

This paper describes the YorkNRM systems submitted to the Trilingual Entity Detection and Linking (EDL) track in 2016 TAC Knowledge Base Population (KBP) contests. Here, we have studied a novel approach for for trilingual entity discovery and mention detection (MD). Instead of treating NER as a sequence labelling problem, we propose to use a new local detection approach, which rely on the recent fixed-size ordinally forgetting encoding (FOFE) method to fully encode each sentence fragment and its left/right contexts into a fixed-size representation. Afterwards, a simple feedforward neural network is used to reject or predict entity label for each individual fragment. Moreover, we have used the entity linking system from another participating team USTC_NELSLIP for entity linking and NIL clustering. Based on these techniques, we have submitted the results to both EDL1 and EDL2 evaluations.

1 Introduction

In this paper, we describe the main techniques we have used to build our entity discovery and mention detection systems for the KBP2016 trilingual EDL tracks. The EDL task requires to detect named entities and their nominal mentions in the raw text of three languages (English, Chinese and Spanish) and further link each detected mention to the corresponding node in an existing knowledge base, namely Freebase. For NIL mentions that do not exist in the knowledge base, the EDL system needs to cluster all NIL mentions and assign a unique ID to each NIL mention cluster.

This year, the EDL task has extended the nominal mention detection to all entity types for all three languages. As before, there are in total 5 different mention types, denoted as PER, LOC, ORG, GPE, FAC. During each evaluation window, a large corpus of 90,000 documents is provided to each team to process. Each EDL system needs to be efficient enough to process these documents within the required evaluation window.

2 Preliminary

In this section, we will briefly review background techniques used in our KBP2016 EDL systems.

2.1 Fixed-size Ordinally Forgetting Encoding

Feedforward neural network is a fast and powerful computation model. However, it requires to use the fixed-size inputs and lacks of the ability to capture long-term dependency in sequences. Because most NLP problems involve variable-length sequences of words, RNNs/LSTMs are more popular than regular feedforward NNs in dealing with these problems. The simple encoding method, called Fixed-size Ordinally Forgetting Encoding (FOFE), originally proposed in (Zhang et al., 2015), nicely overcomes the limitations of DNNs because it can uniquely encode a variable-length sequence of words into a fixed-size representation without losing information.

Give a vocabulary V consisting of $|V|$ distinct words, each word can be represented by a one-hot vector. FOFE mimics bag-of-words (BOW) but incorporates a forgetting factor to capture positional information. It encodes any sequence of variable length composed by words in V. Let $S = w_1, w_2, w_3, ..., w_T$ denote a sequence of T words from V, and $\boldsymbol{e_t}$ be the one-hot vector of the t-th word in S, where $1 \leq t \leq T$. The FOFE of each partial sequence $\boldsymbol{z_t}$ from the first word to the

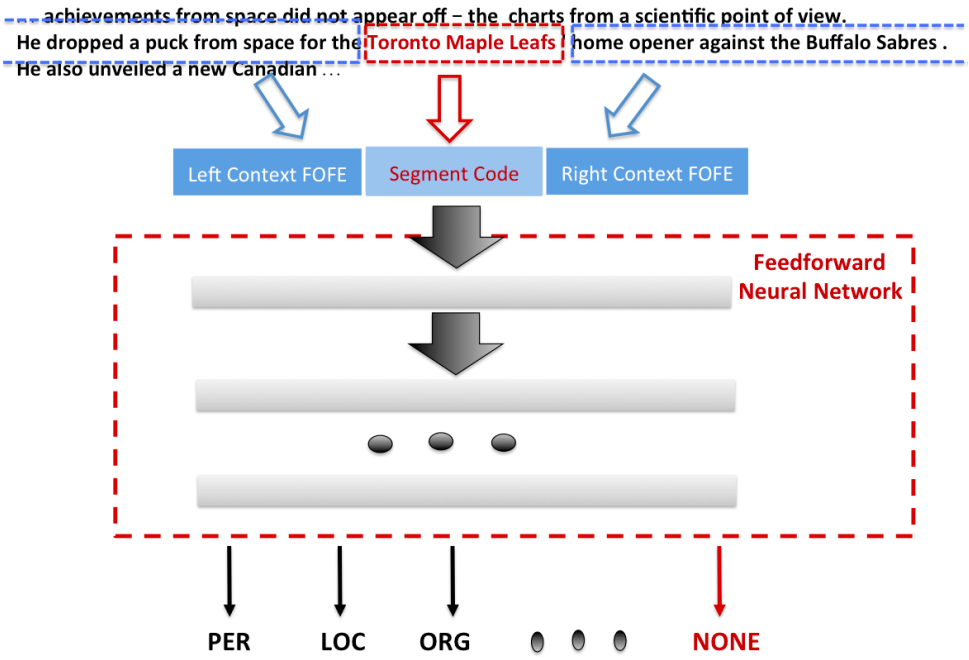

Figure 1: Illustration of the local detection approach for NER using FOFE codes as input and a feedforward neural network as model. The window currently examines the fragment of *Toronto Maple Leafs*. The window will scan and scrutinize all fragments up to K words.

t-th word is recursively defined as:

$$\boldsymbol{z_t} = \begin{cases} \mathbf{0}, & \text{if } t = 0 \\ \alpha \cdot \boldsymbol{z_{t-1}} + \boldsymbol{e_t}, & \text{otherwise} \end{cases} \quad (1)$$

where the constant α is called forgetting factor, and it is chosen between 0 and 1 exclusively. Obviously, the size of $\boldsymbol{z_t}$ is $|V|$, and it is irrelevant to the length of original sequence, T.

Let us use a simple example to illustrate how to use FOFE to decode a sequence. Assume that we have three words in our vocabulary, e.g. A, B, C, whose one-hot representations are $[1, 0, 0]$, $[0, 1, 0]$ and $[0, 0, 1]$ respectively. When calculating from left to right, the FOFE for the sequence "ABC" is $[\alpha^2, \alpha, 1]$ and that of "ABCBC" is $[\alpha^4, \alpha+\alpha^3, 1+\alpha^2]$.

According to (Zhang et al., 2015), FOFE is capable of uniquely encoding any sequence of arbitrary length, serving as a fixed-size but theoretically lossless representation for any sequence.

2.2 Character-level Models in NLP

Recently, as shown in (Kim et al., 2015), it may be beneficial to model morphology in the character level since this may provide some additional advantages in dealing with unknown or out-of-vocabulary (OOVs) words in a language.

The above FOFE method can be easily extended to model character-level feature in NLP. Any word, phrase or fragment can be viewed as a sequence of characters. In this way, based on a pre-defined set of all possible characters, we may apply the same FOFE method to encode the sequence of characters. This always leads to a fixed-size representation, irrelevant to the number of characters in question. For example, a word fragment of "iFLYTEK" may be viewed as a sequence of seven characters: 'i', 'F', 'L', 'Y', 'T', 'E', 'K'. The FOFE codes of this type of character sequences are always fixed-sized and they can be directly fed to a feedforward neural network for morphology modelling.

In the literature, convolutional neural networks (CNNs) have been widely used as character-level models in NLP (Kim et al., 2015).

3 Entity Discovery and Mention Detection

This year we have used a new FOFE-based local detection approach to build our systems for entity discovery and mention detection in KBP2016 trilingual EDL tracks. Our systems, called **FOFE-NER** hereafter, are motivated by the way how human actually infers whether a word segment in

text is an entity or mention, where the entity types of the other entities in the same sentence is not a must. Particularly, the dependency between adjacent entities is fairly weak in NER problems. Whether a fragment is an entity or not, and what class it may belong to, largely depend on the internal structure of the fragment itself as well as the left and right contexts in which it appears. To a large extent, the meaning and spelling of the underlying fragment are informative to distinguish named entities from the rest of the text. Contexts play a very important role in NER or mention detection when it involves multi-sense words/phrases or out-of-vocabulary (OOV) words.

As shown in Figure 1, our proposed **FOFE-NER** method will examine all possible fragments in text (up to a certain length) one by one. For each fragment, it uses the FOFE method to fully encode the underlying fragment itself, its left context and right context into some fixed-size representations, which are in turn fed to a multi-layer feedforward neural network to predict whether the current fragment is not a valid entity mention (*NONE*), or its correct entity type (*PER*, *LOC*, *ORG* and so on). This method is appealing because the FOFE codes serves as a theoretically lossless representation of the hypothesis and its full contexts and the multi-layer neural networks are used as a universal approximator to map from text to the entity labels.

In this work, we use FOFE to explore both word-level and character-level features for each fragment and its contexts.

3.1 Word-level Features

FOFE-NER generates several word-level features for each fragment hypothesis and its left and right contexts as follows:

- Bag-of-word vector of the fragment. For the example in Figure 1, it is a bag-of-word vector of 'Toronto', 'Maple' and 'Leafs'.
- FOFE code for left context including the fragment. In Figure 1, it is the FOFE code of the word sequence of "... *puck from space for the Toronto Maple Leafs* ".
- FOFE code for left context excluding the fragment. In Figure 1, it is the FOFE code of the word sequence of "... *puck from space for the*".
- FOFE code for right context including the fragment. In Figure 1, it is the FOFE code of the word sequence of "... *against opener home ' Leafs Maple Toronto*".
- FOFE code for right context excluding the fragment. In Figure 1, it is the FOFE code of the word sequence of "... *against opener home '* ".

Moreover, all of the above word features are computed for both case-sensitive words in raw text as well as case-insensitive words in normalized lower-case text. These FOFE codes are projected to lower-dimension dense vectors based on two projection matrices, $\mathbf{W}_s$ and $\mathbf{W}_i$, for case-sensitive and case-insensitive FOFE codes respectively. These two projection matrices are initialized by word embeddings trained by *word2vec*, and fine-tuned during the learning of the neural networks.

Due to the recursive computation of FOFE codes in eq.(1), all of the above FOFE codes can be jointly computed for one sentence or document in a very efficient manner.

3.2 Character-level Features

On top of the above word-level features, we also augment character-level features for the underlying segment hypothesis to further model its morphological structure. For the example in Figure 1, the current fragment, *Toronto Maple Leafs*, is considered as a sequence of case-sensitive characters, i.e. "{ 'T', 'o', ..., 'f' , 's' }", we then add the following character-level features for this fragment:

- Left-to-right FOFE code of the character sequence of the underlying fragment. That is the FOFE code of the sequence, "*'T', 'o', ..., 'f' , 's'* ".
- Right-to-left FOFE code of the character sequence of the underlying fragment. That is the FOFE code of the sequence, "*'s' , 'f' , ..., 'o', 'T'* ".

These case-sensitive character FOFE codes are also projected by another character embedding matrix, which is randomly initialized and fine-tuned during model training.

Alternatively, we may use the character CNNs, as described in Section 2.2, to generate character-level features for each fragment hypothesis as well.

3.3 Training and Decoding Algorithm

Obviously, the above **FOFE-NER** model will take each sentence of words, $S = [w_1, w_2, w_3, ..., w_m]$, as input, and examine all continuous sub-sequences $[w_i, w_{i+1}, w_{i+2}, ..., w_j]$ up to n words in S for possible entity types. All sub-sequence longer than n words are considered as non-entity in this work.

When we train the model, based on the entity labels of all sentences in the training set, we will generate many sentence fragments up to n words. These fragments fall into three categories:

- Exact-match with an entity label, e.g., the fragment "*Toronto Maple Leafs*" in the previous example.
- Partial-overlap with an entity label, e.g., "*for the Toronto*".
- Disjoint with all entity label, e.g. "*from space for*".

For all exact-matched fragments, we generate the corresponding outputs based on the types of the matched entities in the training set. For both partial-overlap and disjoint fragments, we introduce a new output label, **NONE**, to indicate that these fragments are not a valid entity. Therefore, the output nodes in the neural networks contains all entity types plus a rejection option denoted as **NONE**.

During training, we implement a produce-consumer software design such that a thread fetches training examples, compute all FOFE codes and packs them as a mini-batch while the other thread feeds the mini-batches to neural networks and adjusts the model parameters and all projection matrices. Since "partial-overlap" and "disjoint" significantly outnumber "exact-match", they are down-sampled so as to balance the data set. During inference, all fragments not longer than n words are all fed to **FOFE-NER** to compute their scores over all entity types. In practice, these fragments can be packed as one mini-batch so that we can compute them in parallel on GPUs. As the NER result, the **FOFE-NER** model will return a subset of fragments only if: i) they are recognized as a valid entity type (not **NONE**); AND ii) The NN scores exceed a global pruning threshold.

Occasionally, some partially-overlapped or nested fragments may occur in the above pruned prediction results. We can use one of the following simple post-processing methods to remove overlappings from the final results:

1. *highest-first*: We check every word in a sentence. If it is contained by more than one fragment in the pruned results, we only keep the one with the maximum NN score and discard the rest.
2. *longest-first*: We check every word in a sentence. If it is contained by more than one fragment in the pruned results, we only keep the longest fragment and discard the rest.

Either of these strategies leads to a collection of non-nested, non-overlapping, non-NONE entity labels.

In some tasks, it may require to label all nested entities. This has imposed a big challenge to the sequence labelling methods. However, the above post-processing can be slightly modified to generate nested entities' labels. In this case, we first run either *highest-first* or *longest-first* to generate the round's result. For every entity survived in this round, we will recursively run either *highest-first* or *longest-first* on all entities in the original set, which are completely contained by it. This will generate more prediction results. This process may continue to allow any levels of nesting. For example, for a sentence of "$w_1\ w_2\ w_3\ w_4\ w_5$", if the model first generates the prediction results after the global pruning, as ["w_2w_3", PER, 0.7], ["w_3w_4", LOC, 0.8], ["$w_1w_2w_3w_4$", ORG, 0.9], if we choose to run *highest-first*, it will generate the first entity label as ["$w_1w_2w_3w_4$", ORG, 0.9]. Secondly, we will run *highest-first* on the two fragments that are completely contained by the first one, i.e., ["w_2w_3", PER, 0.7], ["w_3w_4", LOC, 0.8], then we will generate the second nested entity label as ["w_3w_4", LOC, 0.8]. Fortunately, in any real NER and mention detection tasks, it is pretty rare to have overlapped predictions in the NN outputs. Therefore, the extra expense to run this recursive post-processing method is minimal.

4 Entity Linking and NIL Clustering

In the entity linking task, each detected mention needs to be linked to a known entity in an existing knowledge base, namely Freebase in this task. For all mentions that do not match any existing node in Freebase, we need to cluster these NIL mentions.

This year, we have used a linking system from another participating team, USTC_NELSLIP, (Liu et al., 2016), to conduct entity linking and NIL clustering. Their entity linking (EL) system consists of two modules: a rule based candidate generation and a neural networks probability ranking model. Moreover, some simple string matching rules are used for NIL clustering. See (Liu et al., 2016) for more details.

5 Experimental Results

In KBP2016, the trilingual EDL task is extended to to detect nominal mentions of all 5 entity types for all three languages. In our experiments, for simplicity, we just treat nominal mention types as some extra entity types and detect them along with named entities together with a single model. We have evaluated our proposed FOFE-based local detection method for Entity Discovery in KBP2015 dataset and we have used this method to participate the KBP2016 official tri-lingual EDL evaluation. In the following, we will report the our performance on these KBP EDL tasks.

5.1 Training Data

We make use of the following data sets as our training data to learn the NER and mention detection models.

- **Training and evaluation data in KBP2015**: In previous year's competition, 335 English documents, 313 Chinese documents and 296 Spanish documents were annotated for training and evaluation, totalling 944 documents. In this data set, all five named mention types (PER, ORG, GPE, LOC, FAC) and only one nominal mention type (PER) are labelled. In KBP2016, nominal mention has been expanded to all 5 classes of named entities.

- **Machine-labeled Wikipedia**: When terms or names are first mentioned in a Wikipedia article they are often linked to the corresponding Wikipedia page by hyperlinks, which clearly highlights the possible named entities with well-defined boundary in the text. We have developed a program to automatically map these hyperlinks into KBP annotations by exploring the infobox (if existing) of the destination page and/or examining the corresponding Freebase types. Nominal mentions are not labelled by this approach. In this way, we have created a fairly large amount of weakly-supervised trilingual training data for the KBP2016 EDL task.

- **iFLYTEK's in-house dataset**: The iFLYTEK Research has generously shared with us about 10,000 in-house English and Chinese labeled documents (Liu et al., 2016). These documents are internally labelled by iFLYTEK using some annotation rules similar to the KBP 2016 guidelines.

Additionally, when we generate the machine-labeled data from Wikipedia, we have also created a large gazetteer using the titles of Wikipedia pages and Freebase nodes. We have used the gazetteer-related features for the KBP2016 EDL task.

5.2 Data Preprocessing

Data from both KBP2015 and KBP2016 are in the XML format. Our preprocessing tools only extract text surrounded by two adjacent XML tags for later stages since XML tags tend to be metadata and irrelevant to our task. The values of all author attributes are extracted from all post tags, which are directly labeled as *PER*. The extracted text is sent to the Stanford CoreNLP toolkit for sentence splitting and tokenization. All words containing digits are mapped to several pre-defined tokens, e.g. $\langle number \rangle$, $\langle date \rangle$, using some regular expression matches.

5.3 Hyperparameter optimization

We normally split the available training data into training, validation and evaluation sets in a ratio of 90:5:5. We perform grid search on several hyper-parameters, including initial learning rate, mini-batch size, initial dropout, number of layers, size of hidden layer, number of epochs, on the held-out validation set. Each hyper-parameter typically has 3 to 5 options during the grid search.

Here we summarize the set of hyper-parameters used in our experiments: i) *Number of epochs*: we normally run 256 epochs if the iFLYTEK data is not used in training. Otherwise, we only run 64 epochs. ii) *Learning rate*: it is initially set to 0.128 and it is gradually decreased by multiplying a number at the end of every epoch so that it reaches 1/16 of the initial value at the end of the whole training process; iii) *Dropout rate*: it is initially set to 0.4 and it is slowly decreased

training data	P	R	F_1
KBP2015	0.818	0.600	0.693
KBP2015 + WIKI	0.859	0.601	0.707
KBP2015 + iFLYTEK	0.830	0.652	**0.731**

Table 1: Entity discovery performance (English only) in KBP2016 EDL1 evaluation window is shown as a comparison of three models trained by different combinations of training data sets.

in the training until it reaches 0.1 at the end. iv) *Network structure*: we use a feedforward fully-connected structure of 3 hidden layers, each of which has 512 hidden nodes. The ReLU activation function is used. The network weights are randomly initialized based on a uniform distribution between $-\sqrt{\frac{6}{N_i+N_o}}$ and $\sqrt{\frac{6}{N_i+N_o}}$ (Glorot et al., 2011). v) *Embedding matrices*: case-sensitive and case-insensitive word embeddings for three languages are pre-trained from English Gigaword, Chinese Wikipeida and Spanish Gigaword using the *word2vec* tool (Mikolov et al., 2013). Character embeddings have 128 dimensionare and they are randomly initialized.

5.4 Effect of various training data

In our first set of experiments, we investigate the effect of using different training data sets on the final entity discovery performance. Different training runs are conducted on different combinations of the aforementioned data sources. In Table 1, we have summarized the official English entity discovery results from three systems we submitted to KBP2016 EDL1 evaluation. The first system, using only the KBP2015 data to train the model, has achieved 0.693 in F_1 score in the official KBP2016 English evaluation data. After adding the weakly labelled data, WIKI, we can see the entity discovery performance is improved to 0.707 in F_1 score. Finally, we can see that it yields the best performance by using the KBP2015 data and the iFLYTEK in-house data sets to train our models, giving 0.731 in F_1 score.

5.5 The official performance in KBP2016 EDL evaluation

After fixing some system bugs, we have used both the KBP2015 data and iFLYTEK data to re-train our models for three languages and finally submitted three systems to the final KBP2016 EDL2 evaluation. The official results of two systems are summarized in Table 2. In our systems, we treat all nominal mentions as special types of named entities and both named and nominal entities are recognized using one model. Here we have broken down the system performance according to different languages and categories of entities (named or nominal). In RUN1, we have submitted our best NER system, achieving about 0.718 in F_1 score in the KBP2016 trilingual EDL track. This is a very strong performance among all KBP2016 participating teams. In RUN3, we have submitted system fusion results by combining our results with the best results from another KBP2016 participating team using CNNs and RNNs (Liu et al., 2016). The overall trilingual F_1 score is improved to 0.754. It is worth to note that we have obtained a pretty high recall rate, about 0.735, after the system combination because the NER methods used by these two systems are quite complementary.

At last, using the USTC_NELSLIP entity linking system from (Liu et al., 2016), we have submitted the full EDL results for our run1 during the EDL2 evaluation window. The official trilingual EDL results for the above RUN1 are summarized in Table 3.

6 Conclusions

In this paper, we have described our submitted systems for Trilingual EDL Track of 2016 TAC KBP evaluation. We have investigated a new FOFE-based local detection based approach for the challenging KBP2016 trilingual EDL tasks. This method has relied on the recent fixed-size ordinally forgetting encoding (FOFE) method to fully encode each fragment and its left/right contexts into a fixed-size representation, and a simple feedforward neural network to reject or predict entity label for each individual fragment. Our submitted YorkNRM systems using this method have achieved a strong performance in the official KBP2016 trilingual EDL evaluations.

References

X. Glorot, A. Bordes, and Y. Bengio. 2011. Deep sparse rectifier networks. In *International Conference on Artificial Intelligence and Statistics. JMLR W&CP:*, volume 15, pages 315–323.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2015. Character-aware neural language models. *arXiv preprint arXiv:1508.06615*.

LANG	NAME			NOMINAL			OVERALL		
	P	R	F1	P	R	F1	P	R	F1
	RUN1 (our official ED result in KBP2016 EDL2)								
ENG	0.898	0.789	0.840	0.554	0.336	0.418	0.836	0.680	0.750
CMN	0.848	0.702	0.768	0.414	0.258	0.318	0.789	0.625	0.698
SPA	0.835	0.778	0.806	0.000	0.000	0.000	0.835	0.602	0.700
ALL	0.893	0.759	0.821	0.541	0.315	0.398	0.819	0.639	**0.718**
	RUN3 (system fusion of RUN1 with the best system in (Liu et al., 2016))								
ENG	0.857	0.876	0.866	0.551	0.373	0.444	0.804	0.755	0.779
CMN	0.790	0.839	0.814	0.425	0.380	0.401	0.735	0.760	0.747
SPA	0.790	0.877	0.831	0.000	0.000	0.000	0.790	0.678	0.730
ALL	0.893	0.759	0.821	0.541	0.315	0.398	0.774	**0.735**	**0.754**

Table 2: Official ED performance of our systems on the KBP2016 trilingual EDL2 evaluation.

MEASURE	P	R	F1
strong_all_match	0.721	0.562	**0.632**
typed_mention_ceaf_plus	0.681	0.531	**0.597**

Table 3: The official trilingual linking performance of RUN1 submitted to the EDL2 evaluation window.

Dan Liu, Wei Lin, Shiliang Zhang, Si Wei, and Hui Jiang. 2016. The USTC_NELSLIP systems for trilingual entity detection and linking tasks at TAC KBP 2016. In *Proceedings of Text Analysis Conference (TAC2016)*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Shiliang Zhang, Hui Jiang, Mingbin Xu, Junfeng Hou, and Lirong Dai. 2015. The fixed-size ordinally-forgetting encoding method for neural network language models. In *Proceedings of ACL*.