A FOFE-based Local Detection Approach for Named Entity Recognition and Mention Detection

Anonymous ACL submission

Abstract

In this paper, we study a novel approach for named entity recognition (NER) and mention detection in natural language processing. Instead of treating NER as a sequence labeling problem, we propose 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 (FFNN) is used to reject or predict entity label for each individual fragment. The proposed method has been evaluated in several popular NER and mention detection tasks, including CoNLL 2003 NER task and TAC-KBP2015 and TAC-KBP2016 Tri-lingual Entity Discovery and Linking (EDL) tasks. Our methods have yielded pretty strong performance in all of these examined tasks. This local detection approach has shown many advantages over the traditional sequence labeling methods.

1 Introduction

Natural language processing (NLP) plays an important role in artificial intelligence, which has been extensively studied for many decades. Conventional NLP techniques include the rule-based symbolic approaches widely used about two decades ago, and the more recent statistic approaches that rely on feature engineering and statistical models. In the recent years, deep learning approach has achieved huge successes in many applications, ranging from speech recognition to image classification. It is drawing increasing attention in the NLP community.

In this paper, we are interested in a fundamental problem in NLP, namely named entity recognition (NER) and mention detection (MD). NER and MD is a very challenging task in NLP, laying the foundation of almost every NLP application. NER and MD is a task of identifying entities (named and/or nominal) from raw text, and classifying the detected entities into one of the pre-defined categories such as person, organization, location, etc. Some tasks focus on named entities only, while the others also detects nominal mentions, for example

> $[Mark]_{PER}$ and his closest $[friend]_{PER\_N}$ $[Scarlet]_{PER}$, a cello $[player]_{PER\_N}$, joined the same music $[company]_{ORG\_N}$.

Moreover, nested mentions may need to be extracted too. For example,

> He used to study in $[University\ of\ [Toronto]_{LOC}]_{ORG}$.

where *Toronto* is a LOC entity, embedded in another longer ORG entity *University of Toronto*.

Similar to many other NLP problems, NER and MD is normally formulated as a sequence labeling problem, where a tag is sequentially assigned to each word in the input sentence. It has been extensively studied in the NLP community. The core problem in sequence labeling is to model the conditional probability of an output sequence given an arbitrary input sequence. Many hand-crafted features are combined with statistical models, such as conditional random fields (CRFs), to compute conditional probabilities. More recently, some popular neural networks, including convolutional neural networks (CNNs) and recurrent neural networks (RNNs), are proposed to solve these sequence modeling problems under the popular sequence to sequence modeling framework. The relevant work will be briefly reviewed in Section 2. In the test stage, the learned models compute the conditional probabilities and the output sequence

is generated by the well-known Viterbi decoding algorithm.

In this paper, we propose a novel local detection approach of solving NER and MD problems. The idea can be easily extended to many other sequence labeling problems, such as chunking, part-of-speech tagging (POS). Instead of globally modeling the whole sequence in training and jointly decode the entire output sequence in test, our method examines all word segments (up to a certain length) in a sentence. A word segment will be examined individually based on the underlying segment itself and its left and right contexts in the sentence so as to determine whether this word segment is a valid named entity and the corresponding label if yes. This approach conforms to the way human resolves an NER problem. Given any word fragment and its contexts in a sentence or paragraph, people normally can accurately determine whether this word segment is a named entity or not. People rarely conduct a global decoding over the entire sentence to make such a decision. The key to making an accurate local decision for each individual fragment is to have a full access to the fragment itself as well as its complete contextual information. The main pitfall to implement this idea is that we can not easily encode the segment and its contexts in models since they are of varying lengths in natural languages. Many feature engineering techniques have been proposed but all of these methods will inevitably lead to information loss. In this work, we propose to use a recent fixed-size encoding method, namely fixed-size ordinally forgetting encoding (FOFE) (Zhang et al., 2015), to solve this problem. The FOFE method is a simple recursive encoding method. FOFE theoretically guarantees (almost) unique and lossless encodding of any variable-length sequence. The left and the right contexts for each word segment are encoded by FOFE method, and then a simple neural network can be trained to make a precise recognition for each individual word segment based on the fixed-size presentation of the contextual information. This FOFE-based local detection approach is more appealing to NER and MD. Firstly, feature engineering is almost eliminated. FOFE only relies on a single forgetting factor to encode any sequence. Secondly, under this local detection framework, nested mention is handled with little modification. Next, this local detection approach makes better use of partially-labeled data available from many application scenarios. Sequence labeling model requires all entities in a sentence to be labeled. If only some (not all) entities are labeled, it is not very effective to learn a sequence labeling model. However, every single labeled entity, along with its left and right contexts, may be used to learn the proposed model. At last, due to the flexible encoding strategy of FOFE, simple neural networks, such as multilayer perceptrons, are sufficient for recognition. These models are much faster to train and easier to tune. In the test stage, all possible word segments from a sentence may be packed into a mini-batch, jointly recognized in parallel on GPUs. This leads to a very fast decoding process as well.

In this paper, we have applied this FOFE-based local detection approach to several popular NER and mention detection tasks, including the CoNLL 2003 NER task and TAC-KBP2015 and TAC-KBP2016 Tri-lingual Entity Discovery and Linking (EDL) tasks. Our proposed method has yielded strong performance in all of these examined tasks.

2 Related Work

It has been a long history of research involving neural networks (NN). In this section, we briefly review some recent NN-related research work in NLP, which may be relevant to our work.

The success of word embedding (Mikolov et al., 2013) encourages researchers to focus on machine-learned representation instead of heavy feature engineering in NLP. Using word embedding as the typical feature representation for words, NNs become competitive to traditional approaches in NER. Many NLP tasks, such as NER, chunking and part-of-speech (POS) tagging can be formulated as sequence labeling tasks. In (Collobert et al., 2011), deep convolutional neural networks (CNN) and conditional random fields (CRF) are used to infer NER labels at a sentence level, where they still use many hand-crafted features to improve performance, such as capitalization features explicitly defined based on first-letter capital, non-initial capital and so on.

Recently, recurrent neural networks (RNNs) have demonstrated the ability in modeling sequences (Graves, 2012). Huang et al. (2015) built on the previous CNN-CRF approach by replacing CNNs with bidirectional Long Short-Term Memory (B-LSTM). Though they have reported

improved performance, they employ heavy feature engineering in that work, most of which is language-specific. There is a similar attempt in (Rondeau and Su, 2016), where a full-rank CRF is used. CNNs are used to extract character-level features automatically in (dos Santos et al., 2015).

Gazetteer is a list of names grouped by the predefined categories an NER system is targeting at. Gazetteer is shown to be one of the most effective external knowledge sources to improve NER performance (Sang and Meulder, 2003). Thus, gazetteer is widely used in many NER systems. In (Chiu and Nichols, 2016), state-of-the-art performance on a popular NER task, i.e., CoNLL2003, is achieved by incorporating a large gazetteer. Different from previous ways to use a set of bits to indicate whether a word is in gazetteer or not, they have encoded a match in BIOES (Begin, Inside, Outside, End, Single) annotation, which captures positional information. Their models also make advantage of word embeddings, character-level CNNs and CRFs.

Interestingly enough, none of these recent successes in NER was achieved by a vanilla RNN. Rather, these successes are often established by some the sophisticated models combining CNNs, LSTMs and CRFs in certain ways. In this paper, based on recent work in (Zhang et al., 2015) and (Zhang et al., 2016), we propose a novel but simple solution to NER by applying DNN on top of FOFE-based features. This simpler approach can achieve performance very close to state-of-the-art on various NER and mention detection tasks, without using any external knowledge or feature engineering.

3 Preliminary

In this section, we will briefly review some background techniques, which are important to our proposed NER and mention detection approach.

3.1 Deep Feedforward Neural Networks

It is well known that neural network is a universal approximator under certain conditions (Hornik, 1991). A feedforward neural network (FFNN) is a weighted graph with a layered architecture. Each layer is composed of several nodes. Successive layers are fully connected. Each node applies a function on weighted sum of the lower layer.

Formally, let $x_{n,j}$ denote the value of the j-th node in the n-th layer and $W^n_{i,j}$ denote the weight of the connection from $x_{n,i}$ to $x_{n+1,j}$. Then

$$z_{n+1,j} = \sum_i W^n_{i,j} x_{n,i} \tag{1}$$

$$x_{n+1,j} = \sigma\left(z_{n+1,j}\right) \tag{2}$$

where σ is the activation function, usually chosen to be rectified linear unit (ReLU):

$$\sigma(x) = \max(0, x). \tag{3}$$

For classification tasks, the outputs are normalized into a probability distribution by the so-called *softmax* function, where the i-th node is computed as follows:

$$\sigma(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}. \tag{4}$$

An NN can learn by adjusting its weights in a process called back-propagation. Suppose that we have already calculated the outputs given by an NN for any input. Let $E(y, t)$ be an error metric that measures how incorrect the output y is with respect to the expected target output t. For each weight in NN, we may calculate:

$$\frac{\partial E}{\partial W^n_{i,j}} = \frac{\partial E}{\partial \sigma} \frac{\partial \sigma}{\partial z_{i+1,j}} \frac{\partial z_{i+1,j}}{\partial W^n_{i,j}}. \tag{5}$$

Each weight may be adjusted to slowly reduce this error for each training example, and hence the NN learns to fit the input and the output. This is accomplished by the following the update rule, where α is called the learning rate:

$$W^n_{i,j} := W^n_{i,j} - \alpha \frac{\partial E}{\partial W^n_{i,j}}. \tag{6}$$

The learned NN may be used to generalize and extrapolate to new inputs that have not been seen during training.

3.2 Fixed-size Ordinally Forgetting Encoding

FFNN is a powerful computation model. However, it requires fixed-size inputs and lacks the ability of capturing long-term dependency. Because most NLP problems involves variable-length sequences of words, RNNs/LSTMs are more popular than FFNNs in dealing with these problems. The Fixed-size Ordinally Forgetting Encoding (FOFE), originally proposed in (Zhang et al., 2015), nicely overcomes the limitations of

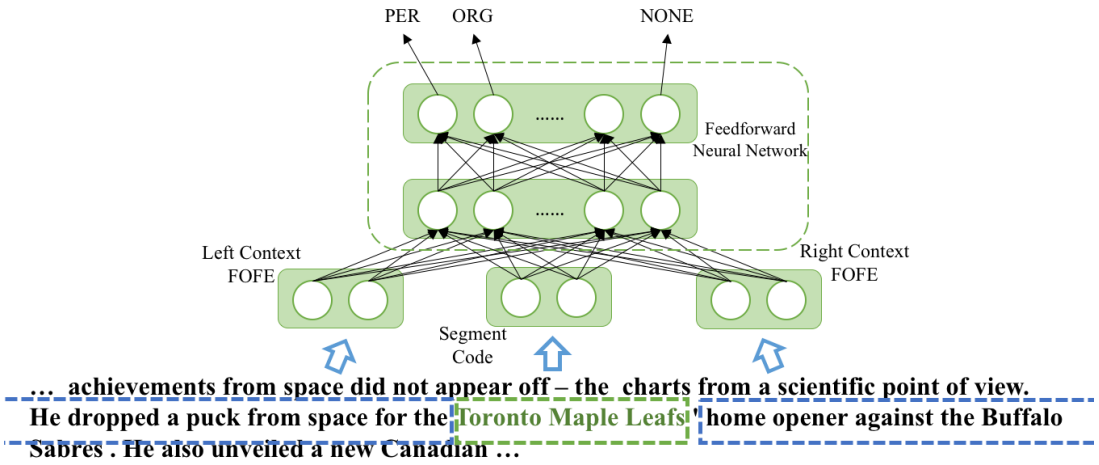

Figure 1: Illustration of the local detection approach for NER using FOFE codes as input and an FFNN as model. The window currently examines the fragment of *Toronto Maple Leafs*. The window will scan and scrutinize all fragments up to K words.

FFNNs because it can uniquely and losslessly encode a variable-length sequence of words into a fixed-size representation.

Give a vocabulary V, 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 t-th word is recursively defined as:

$$\boldsymbol{z_t} = \begin{cases} \boldsymbol{0}, & \text{if } t = 0 \\ \alpha \cdot \boldsymbol{z_{t-1}} + \boldsymbol{e_t}, & \text{otherwise} \end{cases} \quad (7)$$

where the constant α is called forgetting factor, and it is picked 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.

Here's an example. 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 "ABCBC" is $[\alpha^4, \alpha + \alpha^3, 1 + \alpha^2]$.

The word sequences can be unequivocally recovered from their FOFE representations (Zhang et al., 2015). The uniqueness of FOFE representation is theoretically guaranteed. Furthermore, in natural languages, normally a word does not appear repeatedly within a near context.

3.3 Character-level Models in NLP

As shown in (Kim et al., 2016) recently, 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. 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 character sequences are always fixed-sized and they can be directly fed to an FFNN for morphology modeling.

In the literature, convolutional neural networks (CNNs) have been widely used as character-level models in NLP (Kim et al., 2016). Let C denote the set of possible characters, and D denote the dimensionality of character embeddings. A $|C| \times D$ matrix $\boldsymbol{M}$ is randomly initialized, where the i-th row denotes the vector representation of the i-th character in C. Given a word or fragment whose spelling is $[c_1, c_2, c_3, ..., c_L]$, an $L \times D$ matrix $\boldsymbol{C}$ is constructed, where the j-th row is a copy of the row in $\boldsymbol{M}$ corresponding to c_j. $\boldsymbol{C}$ can be viewed as a single-channel image. Let $\boldsymbol{F}$ be an $h \times D$ convolution kernel to be learned, where h denotes

the number of used feature maps. An intermediate vector $\boldsymbol{v}$ of $l-h+1$ elements is generated after $\boldsymbol{f}$ sweeps $\boldsymbol{m}$. Each component in $\boldsymbol{v}$, v_k, is computed as:

$$v_k = \sigma(Trace(\boldsymbol{FC}[k:k+h])) \quad (8)$$

where σ is either sigmoid or ReLU. The output y of this kernel is given by:

$$y = \max(v_1, v_2, ..., v_{l-h+1}) \quad (9)$$

If there are N groups of kernels, each of which has n_1, n_2, n_3, ... , $n_{|N|}$ kernels respectively, following Eqs. (8) and (9), the final representation from the character CNN for this word or fragment is a vector of length $\sum_{i=1}^{|N|} n_i$.

4 FOFE-based Local Detection for NER

As described above, our FOFE-based local detection approach for NER, called **FOFE-NER** hereafter, is 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. 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 an FFNN 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. FFNN is 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.

4.1 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.(7), all of the above FOFE codes can be jointly computed for one sentence or document in a very efficient manner.

4.2 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 3.3, to generate character-level features for each fragment hypothesis as well.

5 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 labeling 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 first round 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.

6 Experiments

In this section, we will evaluate the effectiveness of our proposed methods on several popular NER and mention detection tasks, including CoNLL 2003 NER task and TAC-KBP2015 and TAC-KBP2016 Tri-lingual Entity Discovery and Linking (EDL) tasks. [1]

6.1 CoNLL 2003 NER task

The CoNLL-2003 dataset (Sang and Meulder, 2003) consists of newswire from the Reuters RCV1 corpus tagged with four types of non-nested named entities: location (LOC), organization (ORG), person (PER), and miscellaneous (MISC).

We have investigated the performance of our method on the CoNLL-2003 dataset by using different combinations of the FOFE features (both word-level and character-level). The detailed comparison results are shown in Table 1. In Table 2, we have compared our best performance with some top-performing neural network systems on this task. As we can see from Table 2, our system yields a very strong performance (90.71 in F_1 score) in this task, outperforming most of neural network models reported on this dataset. More importantly, we have not used any hand-crafted features in our systems, and all features (either word or character level) are automatically derived from the data. In (Chiu and Nichols, 2016), a slightly better performance (91.62 in F_1 score) is reported but a customized gazetteer is used in their method.

6.2 KBP2015 EDL Task

Given a document collection in three languages (English, Chinese and Spanish), the KBP2015 trilingual EDL task (Ji et al., 2015) requires to automatically identify entity mentions from a source collection of textual documents in multiple languages, and classify them into one of the following pre-defined five types: Person (PER), Geopolitical Entity (GPE), Organization (ORG), Location (LOC) and Facility (FAC)

As shown in Table 3, our FOFE-based local detection method has obtained fairly strong performance in the KBP2015 dataset. The overall trilingual entity discovery performance is slightly better than the best system participated in the official KBP2015 evaluation, with 73.9 vs. 72.4 as measured by F_1 scores.

6.3 KBP2016 EDL task

In KBP2016, the trilingual EDL task is extended 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 make use of three training data:

- **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 labeled.

- **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 labeled by this approach.

- **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 labeled by iFLYTEK using some annotation rules similar to the KBP 2016 guidelines.

[1] We have made our codes available at https://github.com/user-name/project-name for readers to reproduce the results in this paper.

FEATURE			P	R	F1
word-level	case-insensitive	context FOFE incl. focus word(s)	86.64	77.04	81.56
		context FOFE excl. focus word(s)	53.98	42.17	47.35
		BoW of focus word(s)	82.92	71.85	76.99
	case-sensitive	context FOFE incl. focus word(s)	88.88	79.83	84.12
		context FOFE excl. focus word(s)	50.91	42.46	46.30
		BoW of focus word(s)	85.41	74.95	79.84
char-level	Char FOFE of focus word(s)		67.67	52.78	59.31
	Char CNN of focus word(s)		78.93	69.49	73.91
all case-insensitive features			90.11	82.75	86.28
all case-sensitive features			90.26	86.63	88.41
all word-level features			92.03	86.08	88.96
all word-level & Char FOFE features			91.68	88.54	**90.08**
all word-level & Char CNN features			91.80	88.58	**90.16**
all word-level & all char-level features			93.29	88.27	**90.71**

Table 1: Effect of various FOFE feature combinations on the CoNLL2003 test data.

	word	char	gaz	cap	pos	F1
(Collobert et al., 2011)	✓	✗	✓	✓	✗	89.59
(Huang et al., 2015)	✓	✓	✓	✓	✓	90.10
(Rondeau and Su, 2016)	✓	✗	✓	✓	✓	89.28
(Chiu and Nichols, 2016) [2]	✓	✓	✓	✗	✗	**91.62**
this work	✓	✓	✗	✗	✗	**90.71**

Table 2: Performance (F_1 score) comparison among various neural models reported on the CoNLL dataset, and the different features used in these methods.

	2015 track best			ours		
	P	R	F_1	P	R	F_1
Trilingual	75.9	69.3	72.4	78.3	69.9	**73.9**
English	79.2	66.7	**72.4**	77.1	67.8	72.2
Chinese	79.2	74.8	**76.9**	79.3	71.7	75.3
Spanish	78.4	72.2	75.2	79.9	71.8	**75.6**

Table 3: Entity Discovery Performance of our method on the KBP2015 EDL evaluation data, with comparison to the best system in KBP2015 official evaluation.

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.

6.4 Hyperparameter optimization

We perform grid search on several hyper-parameters. Here we summarize the set of hyper-parameters used in our experiments: i) *Learning rate*: initially set to 0.128 and is multiplied by a decay factor each epoch so that it reaches 1/16 of the initial value at the end of the training; ii) *Network structure*: 3 fully-connected layers of 512 nodes with ReLU activation , 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); iii) *Character embeddings*: 64 dimensionare, randomly initialized. iv) *mini-batch*: 512; v) *Dropout rate*: initially set to 0.4, slowly decreased during training until it reaches 0.1 at the end.

- **KBP series**: i) *Number of epochs*: 64 epochs if the iFLYTEK data is used in training, 64 epochs otherwise; ii) ii) *Embedding matrices*: case-sensitive and case-insensitive word embeddings of 128 dimensions for three languages are pre-trained from English Gigaword, Chinese Wikipeida and Spanish Gigaword using the *word2vec* tool (Mikolov et al., 2013); iii) We normally split the available training data into training, validation and evaluation sets in a ratio of 90:5:5.

- **CoNLL2003**: i) *Number of epochs*: 128;

LANG	NAME			NOMINAL			OVERALL			2016 BEST		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1
ENG	0.898	0.789	0.840	0.554	0.336	0.418	0.836	0.680	0.750	0.846	0.710	0.772
CMN	0.848	0.702	0.768	0.414	0.258	0.318	0.789	0.625	0.698	0.789	0.737	0.762
SPA	0.835	0.778	0.806	0.000	0.000	0.000	0.835	0.602	0.700	0.839	0.656	0.736
ALL	0.893	0.759	0.821	0.541	0.315	0.398	0.819	0.639	**0.718**	0.802	0.704	0.756

Table 4: Official entity discovery performance of our methods on KBP2016 trilingual EDL track.

ii)*Embedding matrices* case-sensitive and case-insensitive word embeddings of 128 dimensions, trained from Reuters RC1; iii) We stick to the official data train-dev-test partition.

6.5 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 5, we have summarized the official English entity discovery results from three systems we submitted to KBP2016 EDL 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 labeled 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 0.750 in F_1 score.

training data	P	R	F_1
KBP2015	0.818	0.600	0.693
KBP2015 + WIKI	0.859	0.601	**0.707**
KBP2015 + iFLYTEK	0.836	0.680	**0.750**

Table 5: Entity discovery performance (English only) in KBP2016 is shown as a comparison of three models trained by different combinations of training data sets.

6.6 The official trilingual performance in KBP2016 EDL

The official results of our system is summarized in Table 4. 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). Our system, achieves 0.718 in F_1 score in the KBP2016 trilingual EDL track, which ranks second in among all participants. Note that our result is produced by a single system while the top system is emsembled by two diffrent models, each of which is from 5-fold cross-validation.

7 Conclusion

In this paper, we have proposed a new local detection based approach, which rely 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. Afterwards, a simple neural network is used to reject or predict entity label for each individual fragment. The proposed method has been evaluated in several popular NER and mention detection tasks, including the CoNLL 2003 NER task and TAC-KBP2015 and TAC-KBP2016 Tri-lingual Entity Discovery and Linking (EDL) tasks. Our methods have yielded pretty strong performance in all of these examined tasks.

References

Jason PC Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics* 4:357–370. https://www.aclweb.org/anthology/Q16-1026.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12(Aug):2493–2537. http://www.jmlr.org/papers/volume12/collobert11a/collobert11a.pdf.

Cıcero dos Santos, Victor Guimaraes, RJ Niterói, and Rio de Janeiro. 2015. Boosting named entity recognition with neural character embeddings. In *Proceedings of NEWS 2015 The Fifth Named Entities Workshop*. Association

for Computational Linguistics (ACL), page 25. https://doi.org/10.18653/v1/w15-3904.

X. Glorot, A. Bordes, and Y. Bengio. 2011. Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics. JMLR W&CP:*. volume 15, pages 315–323. http://www.jmlr.org/proceedings/papers/v15/glorot11a/glorot11a.pdf.

Alex Graves. 2012. Neural networks. In *Supervised Sequence Labelling with Recurrent Neural Networks*, Springer, pages 15–35. https://doi.org/10.1007/978-3-642-24797-2.

Kurt Hornik. 1991. Approximation capabilities of multilayer feedforward networks. *Neural Networks* 4(2):251–257. https://doi.org/10.1016/0893-6080(91)90009-t.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991* https://arxiv.org/abs/1508.01991.

Heng Ji, Joel Nothman, and Ben Hachey. 2015. Overview of tac-kbp2015 tri-lingual entity discovery and linking. In *Proceedings of Text Analysis Conference (TAC2015)*. http://nlp.cs.rpi.edu/paper/kbp2015.pdf.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *AAAI*. Citeseer. https://arxiv.org/abs/1508.06615.

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. https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

Marc-Antoine Rondeau and Yi Su. 2016. LSTM-based NeuroCRFs for named entity recognition. In *Interspeech 2016*. International Speech Communication Association, pages 665–669. https://doi.org/10.21437/interspeech.2016-288.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003,*. page 142147. http://www.aclweb.org/anthology/W03-0419.

Shiliang Zhang, Hui Jiang, Shifu Xiong, Si Wei, and Li-Rong Dai. 2016. Compact feedforward sequential memory networks for large vocabulary continuous speech recognition. In *Interspeech 2016*. International Speech Communication Association. https://doi.org/10.21437/interspeech.2016-121.

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 the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Association for Computational Linguistics (ACL). https://doi.org/10.3115/v1/p15-2081.