A Local Detection Approach for Named Entity Recognition and Mention Detection

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

Abstract

In this paper, we study a novel approach for named entity recognition (NER) and mention detection (MD) in natural language processing. Instead of treating NER as a sequence labeling problem, we propose a new local detection approach, which relies 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. Subsequently, a simple feedforward neural network (FFNN) is learned to either reject or predict entity label for each individual text fragment. The proposed method has been evaluated in several popular NER and MD tasks, including CoNLL 2003 NER task and TAC-KBP2015 and TAC-KBP2016 Tri-lingual Entity Discovery and Linking (EDL) tasks. Our method has 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 statistical approaches relying 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 are very challenging tasks in NLP, laying the foundation of almost every NLP application. NER and MD are tasks 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 (PER), organization (ORG), location (LOC), etc. Some tasks focus on named entities only, while the others also detect nominal mentions. Moreover, nested mentions may need to be extracted too. For example,

$[Sue]_{PER}$ and her $[brother]_{PER\_N}$ studied 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 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 (Borthwick et al., 1998). The core problem 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) (Nguyen et al., 2010), to compute conditional probabilities. More recently, some popular neural networks, including convolutional neural networks (CNNs) and recurrent neural networks (RNNs), are proposed to solve sequence labelling problems. In the inference stage, the learned models compute the conditional probabilities and the output sequence is generated by the Viterbi decoding algorithm (Viterbi, 1967).

In this paper, we propose a novel local detection approach for 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 it is. This approach conforms to the way human resolves an NER problem. Given any word fragment and its contexts in a sentence or paragraph, people 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 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., 2015a,b), to solve this problem. The FOFE method is a simple recursive encoding method. FOFE theoretically guarantees (almost) unique and lossless encoding 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. Secondly, under this local detection framework, nested mention is handled with little modification. Next, it 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 effective to learn a sequence labeling model. However, every single labeled entity, along with its contexts, may be used to learn the proposed model. At last, due to the simplicity 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 MD tasks, including the CoNLL 2003 NER task and TAC-KBP2015 and TAC-KBP2016 Trilingual 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; Liu et al., 2015) 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) with full-rank CRF. CNNs are used to extract character-level features automatically in (dos Santos et al., 2015).

Gazetteer is a list of names grouped by the pre-defined categories. 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.

Interestingly enough, none of these recent successes in NER was achieved by a vanilla RNN. Rather, these successes are often established by sophisticated models combining CNNs, LSTMs and CRFs in certain ways. In this paper, based on recent work in (Zhang et al., 2015a,b) 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 MD 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 the weighted sum of the lower layer. An NN can learn by adjusting its weights in a process called back-propagation. 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., 2015a,b), nicely overcomes the limitations of 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:

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

where the constant α is called forgetting factor, and it is 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 "ABC" is $[\alpha^2, \alpha, 1]$ and that of "ABCBC" is $[\alpha^4, \alpha+\alpha^3, 1+\alpha^2]$.

The word sequences can be unequivocally recovered from their FOFE representations (Zhang et al., 2015a,b). The uniqueness of FOFE representation is theoretically guaranteed by the following two theorems:

Theorem 1. *If the forgetting factor α satisfies $0 < \alpha \leq 0.5$, FOFE is unique for any countable vocabulary V and any finite value T.*

Theorem 2. *For $0.5 < \alpha < 1$, given any finite value T and any countable vocabulary V, FOFE is almost unique everywhere, except only a finite set of countable choices of α.*

Though in theory uniqueness is not guaranteed when α is chosen from 0.5 to 1, in practice the chance of hitting such scenarios is extremely slim, almost impossible due to quantization errors in the system. Furthermore, in natural languages, normally a word does not appear repeatedly within a near context. Simply put, FOFE is capable of uniquely encoding any sequence of arbitrary length, serving as a fixed-size but theoretically lossless representation for any sequence.

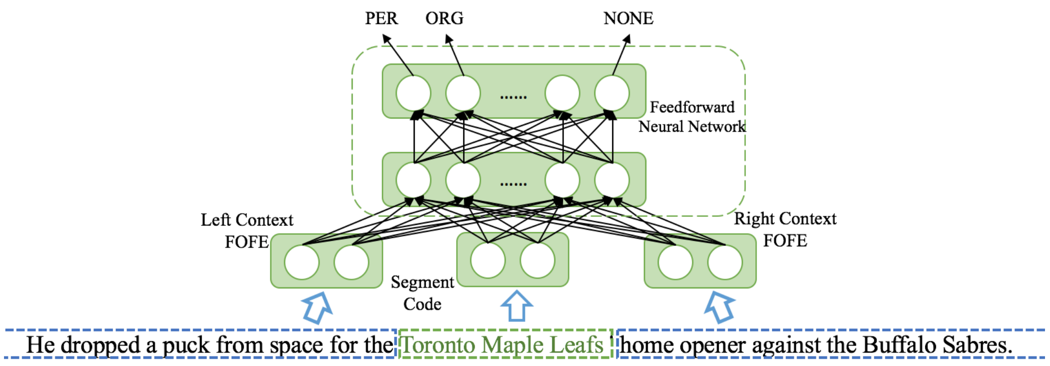
PER
ORG
NONE
Feedforward Neural Network
Left Context FOFE
Segment Code
Right Context FOFE
He dropped a puck from space for the Toronto Maple Leafs home opener against the Buffalo Sabres.

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.

3.3 Character-level Models in NLP

Kim et al. (2016) 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. In the literature, convolutional neural networks (CNNs) have been widely used as character-level models in NLP (Kim et al., 2016). A trainable character embedding is initialized based on a set of possible characters. When a word fragment comes, character vectors are retrieved according to its spelling to construct a matrix. This matrix can be viewed as a single-channel image. CNN is applied to generate a more abstract representation of the word fragment.

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 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 “Walmart” may be viewed as a sequence of seven characters: ‘W’, ‘a’, ‘l’, ‘m’, ‘a’, ‘r’, ‘t’. The FOFE codes of character sequences are always fixed-sized and they can be directly fed to an FFNN for morphology modeling.

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 MD 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) if it is a valid mention. 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 (BoW) of the fragment, e.g.

bag-of-word vector of 'Toronto', 'Maple' and 'Leafs' in Figure 1.

- FOFE code for left context including the fragment, e.g. FOFE code of the word sequence of "... *puck from space for the Toronto Maple Leafs*" in Figure 1.
- FOFE code for left context excluding the fragment, e.g. the FOFE code of the word sequence of "... *puck from space for the*" in Figure 1..
- FOFE code for right context including the fragment, e.g. the FOFE code of the word sequence of "... *against opener home ' Leafs Maple Toronto*" in Figure 1.
- FOFE code for right context excluding the fragment, e.g. the FOFE code of the word sequence of "... *against opener home* " in Figure 1.

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.

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-sequences longer than n words are considered as non-entities 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 producer-consumer software design such that a thread fetches training examples, computes 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) their 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 MD 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 Second-Pass Augmentation

As we know, CRF brings marginal performance gain to all taggers (but not limited to NER) because of the dependancies (though fairly weak) between entity types. We may easily add this level of information to our model by introducing another pass of **FOFE-NER**. We call it *2nd-pass FOFE-NER*.

In 2nd-pass FOFE-NER, another set of model is trained on outputs from the first-pass **FOFE-NER**, including all predicted entities. For example, given a sentence

$$S = [w_1, w_2, ...w_i, ...w_j, ...w_n]$$

and an underlying word segment $[w_i, ..., w_j]$ in the second pass, every predicted entity outside this segment is substituted by its entity type predicted from the first pass. For example, in the first pass, a sentence like *"Google has also recruited Fei-Fei Li, director of the AI lab at Stanford University."* is predicted as: *"<ORG> has also recruited Fei-Fei Li, director of the AI lab at <ORG>."* In 2nd-pass FOFE-NER, when examining the segment "*Fei-Fei Li*", the predicted entity types *<ORG>* are used to replace the actual named entities. The 2nd-pass FOFE-NER model is trained on the outputs of the first pass, where all detected entities are replaced by their predicted types as above.

During inference, the results returned by the 1st-pass model are substituted in the same way. The scores for each hypothesis from 1st-pass model and 2nd-pass model are linear interpolated and then decoded by either *highest-first* or *longest-first* to generate the final results of 2nd-pass FOFE-NER.

Obviously, 2nd-pass FOFE-NER may capture the semantic roles of other entities while filtering out unwanted constructs and sparse combonations. On the other hand, it enables longer context expansion, since FOFE memorizes contextual information in an unselective decaying fashion.

7 Experiments

In this section, we evaluate the effectiveness of our proposed methods on several popular NER and MD tasks, including CoNLL 2003 NER task and TAC-KBP2015 and TAC-KBP2016 Trilingual Entity Discovery and Linking (EDL) tasks.

We have made our codes available at `https://github.com/xmb-cipher/fofe-ner` for readers to reproduce the results in this paper.

7.1 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).

The top 100,000 words, are kept as vocabulary, including punctuations. For the case-sensitive embedding, an OOV is mapped to <unk> if it contains no upper-case letter and <UNK> otherwise. We perform grid search on several hyper-parameters using a held-out dev set. 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 dimensions, 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. vi) *Number of epochs*: 128; vii)*Embedding matrices* case-sensitive and case-insensitive word embeddings of 256 dimensions, trained from Reuters RCV1; viii) We stick to the official data train-dev-test partition. ix) *Forgetting factor* $\alpha = 0.5$. [1]

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 (highest-first decoding) yields very strong performance (90.85 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 char level) are automatically derived from the data. Highest-first and longest-first perform similarly. In (Chiu and Nichols, 2016)[2], a slightly better performance (91.62 in F_1 score) is reported but a customized gazetteer is used in theirs.

7.2 KBP2015 EDL Task

Given a document collection in three languages (English, Chinese and Spanish), the KBP2015 tri-lingual EDL task (Ji et al., 2015) requires to automatically identify entities **(including nested entities)** from a source collection of textual documents in multiple languages as in Table 3, and classify them into one of the following pre-defined five types: Person (PER), Geo-political Entity (GPE), Organization (ORG), Location (LOC) and Facility (FAC). The corpus consists of news articles and discussion forum posts published in recent years, related but non-parallel across languages.

Three models are trained and evaluated independently. Unless explicitly listed, hyperparameters follow those used for CoNLL2003 as described in section 7.1 and 2nd-pass model is not used. Three sets of word embeddings of 128 dimensions are derived from English Gigaword (Parker et al., 2011), Chinese Gigaword (Graff and Chen, 2005) and Spanish Gigaword (Mendonca et al., 2009) respectively. Some language-specific modifications are made:

- **Chinese**: Because Chinese segmentation is not reliable, we label Chinese at character level. The analogous roles of case-sensitive word-embedding and case-sensitive word-embedding are played by character embedding and word-embedding in which the character appears. Neither Char FOFE features nor Char CNN features are used for Chinese.

- **Spanish**: Character set of Spanish is a super set of that of English. When building character-level features, we use the mod function to hash each character's UTF8 encoding into a number between 0 (inclusive) and 128 (exclusive).

As shown in Table 4, our FOFE-based local detection method has obtained fairly strong perfor-

[1]The choice of the forgetting factor α is empirical. We've evaluated $\alpha = 0.5, 0.6, 0.7, 0.8$ on a development set in some early experiments. It turns out that $\alpha = 0.5$ is the best. As a result, $\alpha = 0.5$ is used for all NER/MD tasks throughout this paper.

[2]In their work, they have used a combination of training-set and dev-set to train the model, differing from all other systems (including ours) in Table 2.

FEATURE			P	R	F1
word-level	case-insensitive	context FOFE incl. word fragment	86.64	77.04	81.56
		context FOFE excl. word fragment	53.98	42.17	47.35
		BoW of word fragment	82.92	71.85	76.99
	case-sensitive	context FOFE incl. word fragment	88.88	79.83	84.12
		context FOFE excl. word fragment	50.91	42.46	46.30
		BoW of word fragment	85.41	74.95	79.84
char-level	Char FOFE of word fragment		67.67	52.78	59.31
	Char CNN of word fragment		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**
all features + **dev set** + 5-fold cross-validation			92.58	89.31	**90.92**
all features + **2nd-pass**			92.13	89.61	**90.85**
all features + **2nd-pass** + **dev set** + 5-fold cross-validation			92.62	89.77	**91.17**

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

algorithm	word	char	gaz	cap	pos	F1
CNN-BLSTM-CRF (Collobert et al., 2011)	✓	✗	✓	✓	✗	89.59
BLSTM-CRF (Huang et al., 2015)	✓	✓	✓	✓	✓	90.10
BLSTM-CRF (Rondeau and Su, 2016)	✓	✗	✓	✓	✓	89.28
BLSTM-CRF, char-CNN (Chiu and Nichols, 2016)	✓	✓	✓	✗	✗	**91.62**
Stack-LSTM-CRF, char-LSTM (Lample et al., 2016)	✓	✓	✗	✗	✗	**90.94**
this work	✓	✓	✗	✗	✗	**90.85**

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

	English	Chinese	Spanish	ALL
Train	168	147	129	444
Eval	167	167	166	500

Table 3: Number of Documents in KBP2015

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

mance in the KBP2015 dataset. The overall trilingual entity discovery performance is slightly better than the best systems participated in the official KBP2015 evaluation, with 73.9 vs. 72.4 as measured by F_1 scores. Outer and inner decodings are *longest-first* and *highest-first* respectively.

7.3 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 treat nominal mention types as some extra entity types and detect them along with named entities together with a single model.

7.3.1 Data Description

No official training set is provided in KBP2016. We make use of three sets of training data:

- **Training and evaluation data in KBP2015**: as described in 7.2

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 5: Official entity discovery performance of our methods on KBP2016 trilingual EDL track. Neither KBP2015 nor in-house data labels nominal mentions. Nominal mentions in Spanish are totally ignored since no training data is found for them.

training data	P	R	F_1
KBP2015	0.836	0.598	0.697
KBP2015 + WIKI	0.837	0.628	**0.718**
KBP2015 + in-house	0.836	0.680	**0.750**

Table 6: Our entity discovery official performance (English only) in KBP2016 is shown as a comparison of three models trained by different combinations of training data sets.

- **Machine-labeled Wikipedia (WIKI)**: 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. In this way, we have created a fairly large amount of weakly-supervised trilingual training data for the KBP2016 EDL task. Meanwhile, a gazeteer is created and used in KBP2016.
- **In-house dataset**: A set of 10,000 English and Chinese documents is manually labeled using some annotation rules similar to the KBP 2016 guidelines.

We split the available data into training, validation and evaluation sets in a ratio of 90:5:5. The models are trained for 256 epochs if the in-house data is not used, and 64 epochs otherwise.

7.3.2 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 6, we have summarized the official English entity discovery results from several systems we submitted to KBP2016 EDL evaluation round I and II. The first system, using only the KBP2015 data to train the model, has achieved 0.697 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 0.718 in F_1 score. Moreover, we can see that it yields even better performance by using the KBP2015 data and the in-house data sets to train our models, giving 0.750 in F_1 score.

7.3.3 The official trilingual EDL performance in KBP2016

The official best results of our system are summarized in Table 5. We have broken down the system performance according to different languages and categories of entities (named or nominal). Our system, achieving 0.718 in F_1 score in the KBP2016 trilingual EDL track, ranks second among all participants. Note that our result is produced by a single system while the top system is a combination of two different models, each of which is based on 5-fold cross-validation (Liu et al., 2016).

8 Conclusion

In this paper, we propose a novel solution to NER and MD by applying FFNN on top of FOFE features. This simple local-detection based approach has achieved almost state-of-the-art performance on various NER and MD tasks, without using any external knowledge or feature engineering.

Acknowledgement

This work is supported mainly by a research donation from iFLYTEK Co., Ltd., Hefei, China, and partially by a discovery grant from Natural Sciences and Engineering Research Council (NSERC) of Canada.

References

Andrew Borthwick, John Sterling, Eugene Agichtein, and Ralph Grishman. 1998. Exploiting diverse knowledge sources via maximum entropy in named entity recognition. In *Proc. of the Sixth Workshop on Very Large Corpora*. volume 182. http://ucrel.lancs.ac.uk/acl/W/W98/W98-1118.pdf.

Jason P. C. 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.

David Graff and Ke Chen. 2005. Chinese gigaword. *LDC Catalog No.: LDC2003T09, ISBN* 1:58563–58230.

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.

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

Dan Liu, Wei Lin, Shiliang Zhang, Si Wei, and Hui Jiang. 2016. Neural networks models for entity discovery and linking. *arXiv preprint arXiv:1611.03558* https://arxiv.org/abs/1611.03558.

Quan Liu, Hui Jiang, Si Wei, Zhen-Hua Ling, and Yu Hu. 2015. Learning semantic word embeddings based on ordinal knowledge constraints. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 1501–1511. http://www.aclweb.org/anthology/P15-1145.

Angelo Mendonca, David Andrew Graff, and Denise DiPersio. 2009. *Spanish gigaword second edition*. Linguistic Data Consortium.

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.

Truc-Vien T Nguyen, Alessandro Moschitti, and Giuseppe Riccardi. 2010. Kernel-based reranking for named-entity extraction. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*. Association for Computational Linguistics, pages 901–909. http://www.anthology.aclweb.org/C/C10/C10-2104.pdf.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English gigaword. *Linguistic Data Consortium* .

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.

A. Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory* 13(2):260–269. https://doi.org/10.1109/tit.1967.1054010.

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. 2015a. A fixed-size encoding method for variable-length sequences with its application to neural network language models. *arXiv preprint arXiv:1505.01504*. https://arxiv.org/abs/1505.01504.

Shiliang Zhang, Hui Jiang, Mingbin Xu, Junfeng Hou, and Lirong Dai. 2015b. 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.