An Encoding Strategy Based Word-Character LSTM for Chinese NER

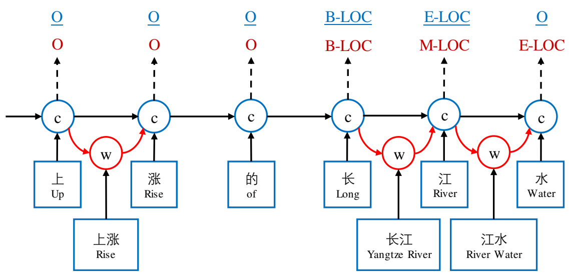
Wei Liu[1,2], Tongge Xu[3,1], Qinghua Xu[1,2], Jiayu Song[2], Yueran Zu[2]
[1]Hefei Innovation Research Institute, Beihang University
[2]School of Computer Science and Engineering, Beihang University
[3]School of Cyber Science and Technology, Beihang University
{liuwei1206, xutg, xuqh_buaa, heavensyc, yueranzu}@buaa.edu.cn

Abstract

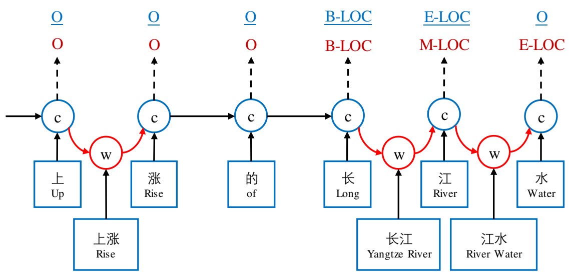
A recently proposed lattice model has demonstrated that words in character sequence can provide rich word boundary information for character-based Chinese NER model. In this model, word information is integrated into a shortcut path between the start and the end characters of the word. However, the existence of shortcut path may cause the model to degenerate into a partial word-based model, which will suffer from word segmentation errors. Furthermore, the lattice model can not be trained in batches due to its DAG structure. In this paper, we propose a novel word-character LSTM(WC-LSTM) model to add word information into the start or the end character of the word, alleviating the influence of word segmentation errors while obtaining the word boundary information. Four different strategies are explored in our model to encode word information into a fixed-sized representation for efficient batch training. Experiments on benchmark datasets show that our proposed model outperforms other state-of-the-arts models.

1 Introduction

Name Entity Recognition(NER) is a basic task of many NLP systems including Information Retrieval (Virga and Khudanpur, 2003), Relationship Extraction (Miwa and Bansal, 2016), Question Answering (Mollá et al., 2006). The main task of NER is to identify named entities such as person, location, organization, etc. in given text. Various methods have been proposed to tackle this problem, including Hidden Markov Models(HMMs) (Saito and Nagata, 2003), Maximum Entropy Models(ME) (Chieu and Ng, 2003), Support Vector Machines(SVM) (Ekbal and Bandyopadhyay, 2010) and Conditional Random Fields(CRF) (Feng et al., 2006). With the development of deep learning, neural net-



(a) Original lattice model



(b) Degraded lattice model in extreme cases

Figure 1: An example of the lattice model degenerates into a partial word-based model. Due to the shortcut path ”江(River)” → ”江水(River Water)” → ”水(Water)”, the model incorrectly predicts that ”江(River)” and ”水(Water)” belong to the same entity. Red labels(without underline) denote predicted labels, and blue labels(with underline) denote gold labels.

works (Huang et al., 2015; Lample et al., 2016; Habibi et al., 2017) have been introduced to NER task. To avoid the segmentation errors, most of neural Chinese NER models are character-based.

Although character-based method has achieved good performance, it does not exploit word information in character sequence. Entity boundaries usually coincide with some word boundaries, which suggests that words in character sequence can provide rich boundary information for character-based model. To integrate words information into character-based model, Zhang and Yang (2018) propose a lattice-structured LSTM model to encode a sequence of input characters as well as all potential words that match a lexicon. Their model is an extension of character-based LSTM-CRF model and uses extra ”shortcut paths” to link the memory cell between the start and the end characters of a word for utilizing word information. And the gated recurrent unit is used to control the contribution of shortcut paths and path between adjacent characters. However, as the study of (Yang et al., 2018) shown, the gate mechanism fails to choose the right path sometimes. As shown in Figure 1, wrong choices may cause lattice model to degenerate into a partial word-based model, which suffers from word segmentation errors. In addition, due to the variable length of words, the length of the whole path is not fixed. Besides, each character is bounded with a variable-sized candidate word sets, which means the amount of incoming and outcoming paths is not fixed either. In this case, lattice LSTM model is deprived of the power of batch training, and hence it is highly inefficient.

To address the above problems, we propose a novel word-character LSTM(WC-LSTM) to integrate word information into character-based model. To prevent our model from degenerating into a partial word-based model, we assign word information to a single character and ensure that there are no shortcut paths between characters. Specifically, word information is assigned to its end character and start character in forward WC-LSTM and backward WC-LSTM respectively. We introduce four strategies to extract fixed-sized useful information from different words, which ensures that our proposed model can perform batch training without losing word information.

We demonstrate the effectiveness of our architecture on four widely used datasets. Experimental results show that our proposed model outperforms other state-of-the-art models on the four datasets.

Our contributions of this paper can be concluded as follows:

- We propose a novel word-character LSTM(WC-LSTM) to incorporate word information into character-based model.

- We explore four different strategies to encode word information into a fixed-sized vector, which enables our proposed model to be trained in batches and adapted to various application scenarios.

- Our proposed model outperforms other models and achieves new state-of-the-art over four Chinese NER datasets. We release the source code for further research[1].

2 Related Work

Neural Networks have been shown to achieve impressive results on Name Entity Recognition task (Gregoric et al., 2018; Lin and Lu, 2018). Based on the level of granularity, most of the models can be divided into three categories: word-based models, character-based models, and hybrid models.

Word-Based Models. Collobert and Weston (2008) propose one of the first word-based models for NER, with feature constructed from orthographic features, dictionaries and lexicons (Yadav and Bethard, 2018). Collobert et al. (2011) replace the hand-crafted features with word embeddings. Huang et al. (2015) propose a BiLSTM-CRF model for NER and achieves good performance. Ma and Hovy (2016) and Chiu and Nichols (2016) use CNN to capture spelling characteristics and Lample et al. (2016) use LSTM instead. When applied to Chinese NER, the above models all suffer from segmentation errors, since Chinese word segmentation is compulsory for those models.

Character-Based Models. Peng and Dredze (2015) propose to add segmentation features for better recognition of entity boundary. Dong et al. (2016) integrate radical-level features into character-based model. To eliminate the ambiguity of character, Sun and He (2017) take the position of character into account. Although the above models have achieved good results, they all ignore word information in character sequence.

Hybrid Models. Some efforts have been made to integrate word boundary information into character-based models. Motivated by the success of multi-task learning for Natural Language Processing (Liu et al., 2016, 2017; Zhang et al., 2018), Peng and Dredze (2016) first proposed to jointly train Chinese NER with Chinese word segmentation(CWS) task. Cao et al. (2018) apply adversarial transfer learning framework to integrate the task-shared word boundary information into Chinese NER task. Another way to obtain word boundary information is proposed by (Zhang and Yang, 2018), using a lattice LSTM to integrate word information into character-based model, which is similar to what is proposed in

[1]https://github.com/liuwei1206/CCW-NER

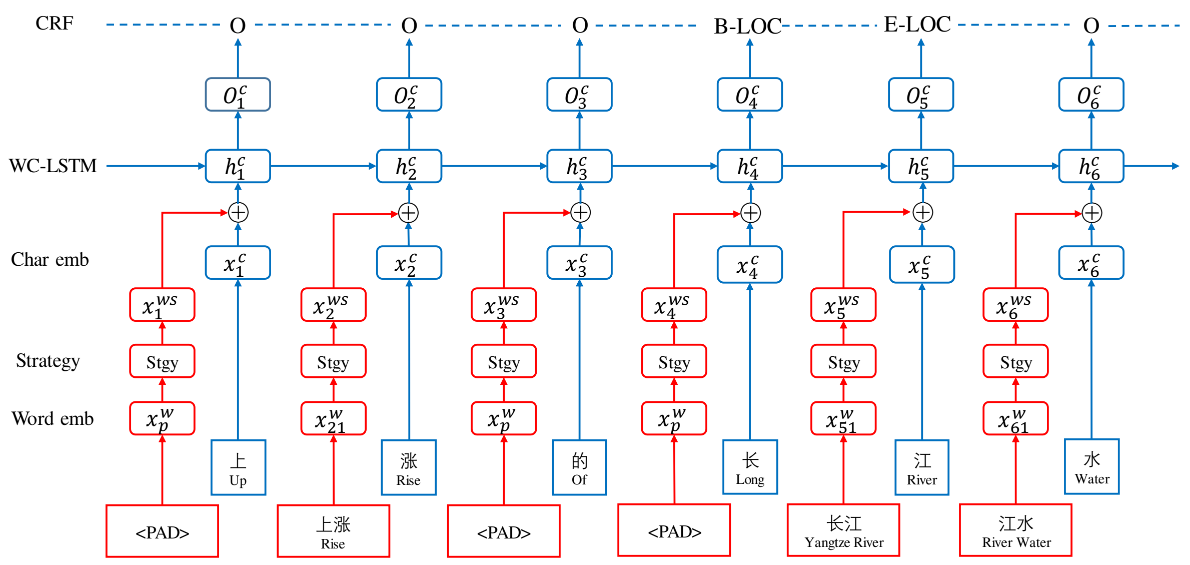


Figure 2: The architecture of our unidirectional model. The blue part can be seen as a standard character-based model but with a word-character LSTM(WC-LSTM), and the red part indicates the process of encoding word information into a fixed-size representation. Word information is integrated into the end character of the word. Where "<PAD>" denotes padding value; "Stgy" denotes a certain encoding strategy and $\oplus$ denotes concatenation operation.

this paper. The main differences are as follows. Firstly, they exploit word information by a DAG-structured LSTM, while we use a chain-structured LSTM. Secondly, instead of integrating to the hidden state of LSTM, our model add word information into the input vector. Finally, our model can be trained in batches and is more efficient.

3 Method

The architecture of our proposed model is shown in Figure 2. Same as the widely used neural Chinese NER model, we use LSTM-CRF as our main network structure. The differences between our model and a standard LSTM-CRF model are mainly on the embedding layer and LSTM and can be summarized as follows. First, we represent a Chinese sentence as a sequence of character-words pairs to integrate word information into each character. Second, to enable our model to train in batches and to meet different application requirements, we introduce four encoding strategies to extract fixed-sized but different information from words. Finally, a chain-structured word-character LSTM is used to extract features from both character and word for better predicting.

Next, we will explain the main ideas for each component, including word-character embedding layer, word encoding strategy, and word-character LSTM.

Formally, we denote a Chinese sentence as $\mathbf{s} = \{c_1, c_2, ..., c_n\}$, where c_i denotes the i_{th} character. We use $c_{b,e}$ to denote a character subsequence in s, which begins with b_{th} character and ends with e_{th} character. Take the sentence in Figure 2 for example, $c_{1,2}$ is "上涨(Rise)". We use $\overrightarrow{ws_i}$ to denote words assigned to i_{th} character in forward WC-LSTM, which are a set of character subsequences $c_{b,i}$, where $b < i$ and $c_{b,i}$ matches a word in lexicon $\mathbf{D}$. The lexicon $\mathbf{D}$ is the same as the one used in (Zhang and Yang, 2018), which is built by using automatically segmented large raw text. Similarly, we use $\overleftarrow{ws_i}$ to denote the words for i_{th} character in backward WC-LSTM, which are a set of character subsequences $c_{i,e}$, where $e > i$ and $c_{i,e}$ matches a word in lexicon $\mathbf{D}$. Finally, the sentence $\mathbf{s}$ is represented as $\overrightarrow{\mathbf{rs}} = \{(c_1, \overrightarrow{ws_1}), (c_2, \overrightarrow{ws_2}), ..., (c_n, \overrightarrow{ws_n})\}$ in our model, and its reverse representation is $\overleftarrow{\mathbf{rs}} = \{(c_n, \overleftarrow{ws_n}), (c_{n-1}, \overleftarrow{ws_{n-1}}), ..., (c_1, \overleftarrow{ws_1})\}$.

3.1 Word-Character Embedding Layer

In our model, Each position i in $\overrightarrow{\mathbf{rs}}$ consists of two parts: i_{th} character c_i and the assigned words $\overrightarrow{ws_i}$. The origin number of words in $\overrightarrow{ws_i}$ is s_i^t, and words are sorted by their length. We ensure each $\overrightarrow{ws_i}$ has the same number s_i^p [2] in the whole batch by padding. We embed each character c_i in dis-

[2]The number depends on the maximum s_i^t in the whole batch, and it can not be less than 1.

tributional space as $\mathbf{x}_i^c$:

$$\mathbf{x}_i^c = \mathbf{e}^c(c_i) \tag{1}$$

where $\mathbf{e}^c$ denotes a pre-trained character embedding lookup table. Similarly, for each $\overrightarrow{ws_i} = \{\overrightarrow{w_{i1}}, ..., \overrightarrow{w_{is_i^p}}\}$, the l_{th} word $\overrightarrow{w_{il}}$ in $\overrightarrow{ws_i}$ is represented using

$$\mathbf{x}_{il}^{\overrightarrow{w}} = \mathbf{e}^w(\overrightarrow{w_{il}}) \tag{2}$$

where $\mathbf{e}^w$ denotes a pre-trained word embedding lookup table. As a result, the distributional representation of words $\overrightarrow{ws_i}$ is $\{\mathbf{x}_{i1}^{\overrightarrow{w}}, ..., \mathbf{x}_{is_i^p}^{\overrightarrow{w}}\}$.

3.2 Words Encoding Strategy

Although the number of assigned words s_i^p for each character c_i is same in one batch, the number varies from batch to batch. As a result, the size of input to the model is not fixed, which is not conducive to batch training. To acquire fixed-sized input, we introduce four different encoding strategies in this section. And we use $\mathbf{x}_i^{\overrightarrow{ws}}$ to denote the final representation of word information for position i in following sections.

Shortest Word First: For each word set $\overrightarrow{ws_i} = \{\overrightarrow{w_{i1}}, ..., \overrightarrow{w_{is_i^p}}\}$, we simply select word whose length is the shortest, i.e. $\overrightarrow{w_{i1}}$. Then

$$\mathbf{x}_i^{\overrightarrow{ws}} = \mathbf{x}_{i1}^{\overrightarrow{w}} \tag{3}$$

Longest Word First: Contrary to the shortest word first, we select word whose length is the longest, i.e. $\overrightarrow{w_{is_i^t}}$. Note that s_i^t may be 0, in this case, we set it to 1. Then

$$\mathbf{x}_i^{\overrightarrow{ws}} = \mathbf{x}_{is_i^t}^{\overrightarrow{w}} \tag{4}$$

Average: While the first two strategies can only use the information of partial words, we introduce an average strategy to utilize all word information. As its name indicates, the average strategy computes the centroid of the embeddings of all elements except paddings in word set , i.e. $\{\overrightarrow{w_{i1}}, ..., \overrightarrow{w_{is_i^t}}\}$. If $s_i^t = 0$, we simply average all the padding value in the word set. Then

$$\mathbf{x}_i^{\overrightarrow{ws}} = \begin{cases} \frac{1}{s_i^t}\sum_{l=1}^{s_i^t}\mathbf{x}_{il}^{\overrightarrow{w}}, & \text{if } s_i^t > 0 \\ \frac{1}{s_i^p}\sum_{l=1}^{s_i^p}\mathbf{x}_{il}^{\overrightarrow{w}}, & \text{if } s_i^t = 0 \end{cases} \tag{5}$$

Self-Attention: Inspired by self-attention mechanism applied to sentence embedding (Lin et al., 2017), we exploit self-attention to better capture useful information from assigned words. For simplicity, we denote all the $\mathbf{x}_{il}^{\overrightarrow{w}}$ as W_i, which has the size s_i^p-by-d^w, where d^w denotes the dimensionality of word embedding $\mathbf{e}^w$.

$$W_i = (\mathbf{x}_{i1}^{\overrightarrow{w}}, ..., \mathbf{x}_{is_i^p}^{\overrightarrow{w}}) \tag{6}$$

We use self-attention mechanism to obtain a linear combination of s_i^p word embeddings in W_i. The attention mechanism takes W_i as input, and generates a weight vector $\mathbf{a}_i$.

$$\mathbf{a}_i = softmax(\mathbf{w}_2 tanh(W_1 W_i^T)) \tag{7}$$

W_1 is a weight matrix with the size of d_a-by-d^w and $\mathbf{w}_2$ is a d_a dimensional vector, where d_a is a hyperparameter. Both of them are trainable parameters.

If $s_i^t > 0$, we use the mask to exclude the padding values; otherwise we reserve them. Finally, we use $\mathbf{a}_i$ to get the weighted sum of all words.

$$\mathbf{x}_i^{\overrightarrow{ws}} = \begin{cases} \sum_{l=1}^{s_i^t} a_{il}\mathbf{x}_{il}^{\overrightarrow{w}}, & \text{if } s_i^t > 0 \\ \sum_{l=1}^{s_i^p} a_{il}\mathbf{x}_{il}^{\overrightarrow{w}}, & \text{if } s_i^t = 0 \end{cases} \tag{8}$$

where a_{il} denotes the l_{th} value in $\mathbf{a}_i$.

3.3 Word-Character LSTM(WC-LSTM)

Inspired by the way character bigram is integrated into sequence labeling model (Chen et al., 2015; Yang et al., 2017), we concatenate each $\mathbf{x}_i^c$ with $\mathbf{x}_i^{\overrightarrow{ws}}$ to utilize word information. And this is quite different from the way used in (Zhang and Yang, 2018), since they use extra shortcut paths to integrate word information into the hidden layer of LSTM. By concatenating, there is no shortcut path in our model and information can only flow between adjacent characters, which ensures that our model will not degenerate into a partial word-based model. Then the WC-LSTM functions are:

$$\begin{bmatrix} \tilde{\mathbf{c}}_i \\ \mathbf{o}_i \\ \mathbf{i}_i \\ \mathbf{f}_i \end{bmatrix} = \begin{bmatrix} tanh \\ \sigma \\ \sigma \\ \sigma \end{bmatrix} \left(\mathbf{W}_p \begin{bmatrix} \mathbf{x}_i \\ \mathbf{h}_{i-1} \end{bmatrix} + \mathbf{b}_p \right) \tag{9}$$

$$\begin{aligned} \mathbf{x}_i &= \mathbf{x}_i^c \oplus \mathbf{x}_i^{\overrightarrow{ws}} \\ \mathbf{c}_i &= \tilde{\mathbf{c}}_i \odot \mathbf{i}_i + \mathbf{c}_{i-1} \odot \mathbf{f}_i \\ \mathbf{h}_i &= \mathbf{o}_i \odot tanh(\mathbf{c}_i) \end{aligned} \tag{10}$$

where $\mathbf{o}_i$, $\mathbf{i}_i$ and $\mathbf{f}_i$ denote output gate, input gate and forget gate respectively. $\mathbf{W}_p$ and $\mathbf{b}_p$ are parameters of affine transformation; σ denotes the logistic sigmoid function; $\oplus$ denotes concatenation operation and $\odot$ denotes elementwise multiplication.

The bidirectional WC-LSTM is applied in our model to leverage both information from the past and the future. To get the future information, we use a second WC-LSTM that reads the reverse representation of $\overrightarrow{\mathbf{rs}}$, i.e., $\overleftarrow{\mathbf{rs}} = \{(c_n, \overleftarrow{ws_n}), (c_{n-1}, \overleftarrow{ws_{n-1}}), ..., (c_1, \overleftarrow{ws_1})\}$. And the following operations to get each backward WC-LSTM hidden vector $\overleftarrow{\mathbf{h}_i}$ is the same as the one in the forward WC-LSTM. Finally, the update of each bidirectional WC-LSTM unit can be written as follows:

$$\begin{aligned}
\overrightarrow{\mathbf{x}_i} &= \mathbf{x}_i^c \oplus \mathbf{x}_i^{\overrightarrow{ws}} \\
\overleftarrow{\mathbf{x}_i} &= \mathbf{x}_i^c \oplus \mathbf{x}_i^{\overleftarrow{ws}} \\
\overrightarrow{\mathbf{h}_i} &= \overrightarrow{\mathrm{WC-LSTM}}(\overrightarrow{\mathbf{h}_{i-1}}, \overrightarrow{\mathbf{x}_i}) \\
\overleftarrow{\mathbf{h}_i} &= \overleftarrow{\mathrm{WC-LSTM}}(\overleftarrow{\mathbf{h}_{i+1}}, \overleftarrow{\mathbf{x}_i}) \\
\mathbf{h}_i &= \overrightarrow{\mathbf{h}_i} \oplus \overleftarrow{\mathbf{h}_i}
\end{aligned} \tag{11}$$

where $\overrightarrow{\mathbf{h}_i}$ and $\overleftarrow{\mathbf{h}_i}$ are hidden states at position i of forward and backward WC-LSTM respectively, and $\oplus$ denotes concatenation operation.

3.4 Decoding and Training

Considering the dependencies between successive labels, we use a CRF layer to make sequence tagging. We define matrix $\mathbf{O}$ to be scores calculated based on the output $\mathbf{H} = \{\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n\}$:

$$\mathbf{O} = \mathbf{W}_o\mathbf{H} + \mathbf{b}_o \tag{12}$$

For a label sequence $\mathbf{y} = \{y_1, y_2, ..., y_n\}$, we define its probability to be:

$$p(\mathbf{y}|\mathbf{s}) = \frac{exp\left(\sum_i \left(\mathbf{O}_{i,y_i} + \mathbf{T}_{y_{i-1},y_i}\right)\right)}{\sum_{\tilde{\mathbf{y}}} exp\left(\sum_i \left(\mathbf{O}_{i,\tilde{y}_i} + \mathbf{T}_{\tilde{y}_{i-1},\tilde{y}_i}\right)\right)} \tag{13}$$

Where $\mathbf{W_o}$ and $\mathbf{b_o}$ are paramters to calculate $\mathbf{O}$; $\mathbf{T}$ is a transition score matrix and $\tilde{\mathbf{y}}$ denotes all possible tag sequences.

While decoding, we use the Viterbi algorithm to find the label sequences that obtained the highest score:

$$\mathbf{y}^* = arg \max_{\mathbf{y} \in \tilde{\mathbf{y}}} \sum_i \left(\mathbf{O}_{i,y_i} + \mathbf{T}_{y_{i-1},y_i}\right) \tag{14}$$

Given N manually labeled data $\{(\mathbf{s}_j, \mathbf{y}_j)\}|_{j=1}^N$, we minimize the sentence-level negative log-likelihood loss to train the model:

$$L = -\sum_j log(p(\mathbf{y}_j|\mathbf{s}_j)) \tag{15}$$

Dataset	Train sent	Dev sent	Test sent
OntoNotes	15724	4301	4346
MSRA	46364	-	4365
Weibo NER	1350	270	270
Chinese resume	3821	463	477

Table 1: Statistics of the datasets

4 Experiments

4.1 Experimental Settings

Dataset. We evaluate our model on four datasets, including OntoNotes4 (Weischedel et al., 2011), MSRA (Levow, 2006), Weibo NER (Peng and Dredze, 2015) and a Chinese resume dataset (Zhang and Yang, 2018). Both OntoNotes4 and MSRA datasets are news in simplified Chinese. Weibo NER dataset is social media data, which is drawn from the Sina Weibo. Chinese resume dataset consists of resumes of senior executives, which is annotated by (Zhang and Yang, 2018). For OntoNotes, we use the same training, development and test splits as (Che et al., 2013). For other datasets which have already been split, and we don't change them. We summarize the datasets in Table 1.

Implementation Details. We utilize the character and word embeddings used in (Zhang and Yang, 2018), both of which are pre-trained on Chinese Giga-Word using word2vec model. Following (Zhang and Yang, 2018), we use the word embedding dictionary as Lexicon $\mathbf{D}$ in our model. For characters and words that do not appear in the pretrained embeddings, we initialize them with a uniform distribution[3]. When training the model, character embeddings and word embeddings are updated along with other parameters.

For hyper-parameter configurations, we mostly refer to the settings in (Zhang and Yang, 2018). We set both character embedding size and word embedding size to 50. The dimensionality of each unidirectional multi-input LSTM hidden states is 100 for Weibo NER and Chinese Resume, and 200 for OntoNote 4 and MSRA. For self-attention strategy, we set the d_a to 50. To avoid overfitting, we apply dropout to both embeddings and LSTM with a rate of 0.5. We use SGD to optimize all the trainable parameters. Learning rate is set to 0.015 initially and decays during training at a rate

[3]The range is $\left[-\sqrt{\frac{3}{dim}}, +\sqrt{\frac{3}{dim}}\right]$, where dim demotes the size of embedding.

Input	Models	P	R	F1
Gold seg	Wang et al. (2013)	76.43	72.32	74.32
	Che et al. (2013)	**77.71**	72.51	**75.02**
	Yang et al. (2016)	65.59	71.84	68.57
No seg	Lattice (Zhang and Yang, 2018)	76.35	71.56	73.88
	Character baseline	70.08	60.53	64.95
	WC-LSTM + shortest	**76.39**	72.39	**74.34**
	WC-LSTM + longest	75.62	72.76	**74.16**
	WC-LSTM + average	76.04	72.03	**73.98**
	WC-LSTM + self-attention	76.09	**72.85**	**74.43**

Table 2: Results on OntoNotes

of 0.05.

For evaluation, we use the Precision(P), Recall(R) and F1 score as metrics in our experiments.

4.2 Experimental Results

OntoNotes. Table 2 shows the experimental results on OntoNote 4 dataset. The ”Input” column shows the representation of input sentence, where ”Gold seg” means a sequence of words with gold-standard segmentation, and ”No seg” means a sequence of character without any segmentation.

The first block in Table 2 are the results of word-based models (Wang et al., 2013; Che et al., 2013; Yang et al., 2016). By using gold-standard segmentation and external labeled data, all of them achieve good performance. But the only resource used in our model are pretrained character and word embeddings.

The first two rows in the second block show the performance of the lattice model and character-based model. The character baseline denotes the original character-based BiLSTM-CRF model. Zhang and Yang (2018) propose a lattice LSTM to exploit word information in character sequence, giving the F1 score of 73.88%. Compared with the character baseline, lattice model gains 8.92% improvement in F1 score, which shows the importance of word information in character sequence.

In the last four rows, we list the results of our proposed model. The results show that all of our models outperform other character-based models, and the one with self-attention strategy achieves the best result. Without gold-standard segmentation and external labeled data, our model gives competitive results to the word-based models on this dataset. Compared with the character baseline, our model with self-attention obtains 9.48% improvement in F1 score, which proves the effectiveness of our way to integrating word information. Compared with lattice model, all of our models achieve better results, which shows that our approach to integrating word information is more reasonable than lattice model.

Models	P	R	F1
Zhang et al. (2006)	92.20	90.18	91.18
Zhou et al. (2013)	91.86	88.75	90.28
Dong et al. (2016)	91.28	90.62	90.95
Cao et al. (2018)	91.73	89.58	90.64
Lattice (Zhang and Yang, 2018)	93.57	92.79	93.18
Character baseline	89.61	86.98	88.37
WC-LSTM + shortest	93.97	92.59	**93.28**
WC-LSTM + longest	94.33	**93.11**	**93.71**
WC-LSTM + average	**94.58**	92.91	**93.74**
WC-LSTM + self-attention	94.36	92.38	**93.36**

Table 3: Results on MSRA

Models	NE	NM	Overall
Peng and Dredze (2015)	51.96	61.05	56.05
Peng and Dredze (2016)	**55.28**	62.97	58.99
Sun and He (2017)	54.50	62.17	58.23
He and Sun (2017)	50.60	59.32	54.82
Cao et al. (2018)	54.34	57.35	58.70
Lattice (Zhang and Yang, 2018)	53.04	62.25	58.79
Character baseline	47.98	57.94	52.88
WC-LSTM + shortest	52.99	65.75	**59.20**
WC-LSTM + longest	52.55	**67.41**	**59.84**
WC-LSTM + average	**53.19**	64.17	58.67
WC-LSTM + self-attention	49.86	65.31	57.51

Table 4: Results on Weibo NER

MSRA. Table 3 shows the results on MSRA dataset. Zhang et al. (2006) and Zhou et al. (2013) use the statistical model with rich hand-crafted features. Dong et al. (2016) exploit radical features in Chinese character. Cao et al. (2018) joint train Chinese NER task with Chinese word segmentation, in which adversarial learning and self-attention mechanism are applied for better performance. We can observe that our proposed models outperformance the above models and the one with average strategy achieves new state-of-the-art performance.

Weibo. Table 4 shows the results[4] on Weibo dataset. The ”NE”, ”NM” and ”Overall” columns denote F1-score for named entities, nominal entities(excluding named entities) and both respectively. We can see that WC-LSTM model with longest word first strategy achieves new state-of-the-art performance. Multi-task learning (Peng and Dredze, 2015, 2016; Cao et al., 2018) and semi-supervised learning (Sun and He, 2017; He and Sun, 2017) are the most common methods

[4]The results of (Peng and Dredze, 2015, 2016) are taken from (Peng and Dredze, 2017)

Models	P	R	F1
Lattice (Zhang and Yang, 2018)	94.81	94.11	94.46
Character baseline	93.26	93.44	93.35
WC-LSTM + shortest	94.97	94.91	**94.94**
WC-LSTM + longest	**95.27**	**95.15**	**95.21**
WC-LSTM + average	95.09	94.97	**95.03**
WC-LSTM + self-attention	95.14	94.79	**94.96**

Table 5: Results on Chinese Resume

	Time(s)/epoch
Character baseline (*batch_size*=1)	880
Character baseline (*batch_size*=8)	253
Lattice	2245
WC-LSTM (*batch_size*=1)	980
WC-LSTM (*batch_size*=8)	350

Table 6: Time per epoch of models

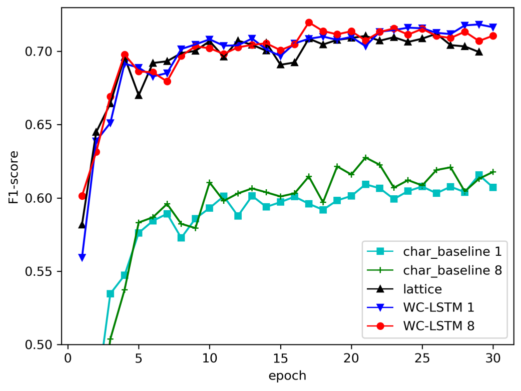


Figure 3: Convergence curve of models. Our model can converge within the same epochs as lattice model does. ”1” and ”8” denotes batch_size. Lattice model can only be trained with batch_size=1 due to its DAG structure.

for Weibo NER task due to the small amount of training data. All of the above models require additional cross-domain or semi-supervised data. Compared with those models, our model does not need additional labeled data; we only exploit pre-trained character and word embeddings.

Resume. Table 5 shows the results on Chinese Resume dataset. Consistent with the previous results, our models outperform lattice model (Zhang and Yang, 2018). The above experimental results strongly verify that our method to utilize word information is more effective than the lattice model.

Our proposed model has achieved state-of-the-art results on **various domains** such as news, social media, and Chinese resume.

4.3 Efficiency

To further explore the efficiency of our model, we conduct some comparative experiments on training time and convergence speed. The lattice model proposed in (Zhang and Yang, 2018) is our principal comparison object, since it also utilizes the word information in character sequence. Our model is an extension of the character-based model, so we also report the results on character-based model as character baseline. We only conduct our experiments on OnteNotes dataset due to space limitation. And we choose the model with the self-attention strategy for the comparative experiments, as it outperforms other strategies on OntoNotes dataset.

The training time of each epoch for all models is shown in Table 6. The lattice model needs the most training time for each epoch, since it can only be trained with batch_size=1 due to its complex DAG structure. Compared with it, our model with batch_size=1 only need half of the training time. Which shows that our model is more efficient. With batch_size=8, our model is nearly 6 times faster than the lattice model, which further demonstrates the efficiency of our model. Compared with the character baseline, our model only adds a small amount of training time but greatly improves the performance. All the experiments are conducted on a single GPU with NVIDIA Tesla K40m.

Figure 3 shows the learning curve of the models in Table 6. As we can see from the figure, whether with batch_size=1 or 8, our model can converge within the same epochs as lattice model does. But compared with the lattice model, our model with batch_size=8 only takes about 1/7 of their training time per epoch. Besides, we can observe from Figure 3, both our model and lattice model significantly outperform the character baseline, which shows the importance of the word information again.

4.4 Detailed Analysis

Case Study. Word information is very useful for Chinese NER task, since it can provide rich word boundary information. To verify that our model can better utilize the boundary information, we analyze an example from OntoNotes dataset. As shown in Table 7, the character-based model cannot detect the existence of the entity ”东北亚(Northeast Asia)” without word information. The lattice model incorrectly recognizes ”东北亚

Sentence (truncated)	新的 New		东北亚 Northeast Asian			大陆桥 Continental Bridge		
Latent words	东北, 东北亚, 北亚, 亚大, 亚大陆, 大陆, 大陆桥, 陆桥 Northest, Northeast Asia, North Asia, Second largest, Subcontinent, Continent, Continental bridge, Land bridge							
Gold labels	新	的	东	北	亚	大	陆	桥
	O	O	B-LOC	M-LOC	E-LOC	O	O	O
Character	新	的	东	北	亚	大	陆	桥
	O	O	O	O	O	O	O	O
Lattice	新	的	东	北	亚	大	陆	桥
	O	O	B-LOC	M-LOC	M-LOC	M-LOC	M-LOC	E-LOC
Shortest	新	的	东	北	亚	大	陆	桥
	O	O	B-LOC	M-LOC	E-LOC	O	O	O
Longest	新	的	东	北	亚	大	陆	桥
	O	O	B-LOC	M-LOC	E-LOC	O	O	O
Average	新	的	东	北	亚	大	陆	桥
	O	O	B-LOC	M-LOC	E-LOC	O	O	O
Self-attention	新	的	东	北	亚	大	陆	桥
	O	O	B-LOC	M-LOC	E-LOC	O	O	O

Table 7: An example of that our models can mitigate the influence of wrong boundary information while utilizing word information. ”Latent words” denotes all words in character sequences; ”Character” denotes the character-based model; ”Lattice” denotes lattice model and the last four rows are our models with different encoding strategies.

Sentence (truncated)	房维中经叔平 Fang Weizhong and Jing Shuping					
Latent words	房维, 中经, 经叔平, 经叔, 叔平 Fang Wei, Zhongjing, Jing Shuping, Jingshu, Shuping					
Gold labels	房	维	中	经	叔	平
	B-PER	M-PER	E-PER	B-PER	M-PER	E-PER
Average	房	维	中	经	叔	平
	B-PER	M-PER	E-PER	B-PER	M-PER	E-PER
Self-attention	房	维	中	经	叔	平
	B-PER	M-PER	M-PER	E-PER	B-PER	E-PER

Table 8: An example of our model applied to informal text using the average strategy and self-attention strategy. ”Latent words” denotes all words in character sequences.

大陆桥(Northeast Asian Continental Bridge)” as an entity, which is caused by the wrong selection of paths. Different from the lattice model, our models are not disturbed by the wrong boundary information and make the correct predictions.

Strategies Analysis. In this part, we analyze the difference between strategies. The application scenarios of shortest word first and longest word first can be explained by Nested Name Entity Recognition (Ju et al., 2018; Sohrab and Miwa, 2018). Short word first is good at identifying inner nested entities due to the short word information, while longest word first tends to identify flat entities with the help of long word information. Taking ”长江三角洲(Yangtze River Delta)” as an example, shortest word first recognizes ”长江(Yangtze)” and ”三角洲(Delta)” as entities, but longest word first tend to think that they are part of the entity ”长江三角洲(Yangtze River Delta)”. Both results are reasonable, but the right result depends on specific needs.

The average and self-attention strategies are the combination of all words information and can use more information. Intuitively, they should outperform the shortest word first and the longest word first. But results on Weibo NER(Table 4) and Resume(Table 5) show the opposite effect. We conjecture that this is caused by the small amount of training data since more word information but small dataset will lead to overfitting. The average strategy is a special case of the self-attention strategy where all weights are the same, so we would like to see the latter outperforms the former when training data is sufficient. Surprisingly, the average strategy achieves higher F1 score than the self-attention strategy in MSRA dataset(Table 3). We carefully analyze the experimental results and find that there are a large number of informal texts in the MSRA test set. Specifically, the MSRA test set contains some very long sentences, in which there are a series of Chinese person name without delimiter. As shown in Table 8, when applied to such informal text, the self-attention strategy fails to determine the entity boundary sometimes while the average strategy correctly recognizes the entities. And we conjecture that, with more trainable parameters, the self-attention strategy can better fit the formal text in the training set but cannot adapt well to the informal data in the test set, so it performs worse than the average strategy.

Finally, the application scenarios of different strategies can be summarized as followings. If the training data is sufficient, we recommend using self-attention for formal texts and average strategy for informal texts. If there is only a very small amount of annotated data, we recommend using the shortest words first for inner nested entities and longest word first strategy for flat entities.

Lexicon and Embeddings. To further analyze the contribution from word lexicon and pretrained word embeddings, we conduct some comparative experiments by using the same word lexicon with and without pretrained embeddings. We choose the strategy that achieving the best performance for each dataset. We estimate the contribution of the lexicon by replacing pretrained word embeddings with randomly initialized embeddings[5]. As shown in Table 9, both lexicon

[5]Same initialization strategy as in ”Implement Details”.

	OntoNotes	Resume	MSRA	Weibo
Character baseline	64.95	93.35	88.37	52.88
WC-LSTM + init	67.81(+2.86)	94.51(+1.16)	90.68(+2.31)	54.94(+2.06)
WC-LSTM + pretrain	74.43(+9.48)	95.21(+1.86)	93.74(+5.37)	59.84(+6.96)

Table 9: Comparison F1 scores between our proposed model with and without pretrained word embeddings. Where "init" and "pretrain" denote without and with pretrained embeddings respectively. "+" denotes the boost value to baseline.

and pretrained word embeddings are useful to our model. However, different from the result in lattice model(Yang et al., 2018), pretrained word embeddings contribute more than lexicon to our model. Taking the result on Ontonote for example, the contribution of pretrained embeddings can be estimated as $(9.48\% - 2.86\%) = 6.62\%$, which is higher than the contribution of lexicon 2.86%. The results show that our model relies more on pretrained embeddings instead of the lexicon, which explains the excellent performance of our model in different domains.

5 Conclusion and Future Work

In this paper, we propose a novel method to utilize word information in character sequence for Chinese NER. Four encoding strategies are introduced to extract fixed-sized but different information for batch training. By using WC-LSTM to extract features from the character vector and word vector, our model can effectively exploit word boundary information and mitigate the influence of word segmentation errors. Experiments on datasets in different domains show that our model is more efficient and faster than the lattice model and also outperforms other state-of-the-art models.

In the future, we plan to further improve and perfect the proposed method, such as exploring some strategies to handle OOV words. Also, the proposed methods can be further extended to other Chinese NLP tasks, such as CWS, Text Classification, and Sentiment Analysis.

Acknowledgments

The research work is supported by the No. BHKX-17-07 project of Hefei Innovation Research Institute, Beihang University. We would like to thank Jie Yang for his open-source toolkit "NCRF++" (Yang and Zhang, 2018). Moreover, we sincerely thank all anonymous reviewers for their valuable comments.

References

Pengfei Cao, Yubo Chen, Kang Liu, Jun Zhao, and Shengping Liu. 2018. Adversarial transfer learning for chinese named entity recognition with self-attention mechanism. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 182–192.

Wanxiang Che, Mengqiu Wang, Christopher D. Manning, and Ting Liu. 2013. Named entity recognition with bilingual constraints. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, pages 52–62.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. 2015. Long short-term memory neural networks for chinese word segmentation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1197–1206.

Hai Leong Chieu and Hwee Tou Ng. 2003. Named entity recognition with a maximum entropy approach. In *Proceedings of the Seventh Conference on Natural Language Learning, CoNLL 2003, Held in cooperation with HLT-NAACL 2003, Edmonton, Canada, May 31 - June 1, 2003*, pages 160–163.

Jason P. C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *TACL*, 4:357–370.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008)*, pages 160–167.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12.

Chuanhai Dong, Jiajun Zhang, Chengqing Zong, Masanori Hattori, and Hui Di. 2016. Character-based LSTM-CRF with radical-level features for chinese named entity recognition. In *In Natural Language Understanding and Intelligent Applications*, pages 239–250.

Asif Ekbal and Sivaji Bandyopadhyay. 2010. Named entity recognition using support vector machine: A language independent approach. *International Journal of Electrical and Electronics Engineering*, 4(2):155–170.

Yuanyong Feng, Le Sun, and Yuanhua Lv. 2006. Chinese word segmentation and named entity recognition based on conditional random fields models. In

Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing, pages 181–184.

Andrej Zukov Gregoric, Yoram Bachrach, and Sam Coope. 2018. Named entity recognition with parallel recurrent neural networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 69–74.

Maryam Habibi, Leon Weber, Mariana L. Neves, David Luis Wiegandt, and Ulf Leser. 2017. Deep learning with word embeddings improves biomedical named entity recognition. *Bioinformatics*, 33(14):i37–i38.

Hangfeng He and Xu Sun. 2017. A unified model for cross-domain and semi-supervised named entity recognition in chinese social media. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3216–3222.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *Computing Research Repository*, arXiv:1508.01991.

Meizhi Ju, Makoto Miwa, and Sophia Ananiadou. 2018. A neural layered model for nested named entity recognition. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1446–1459.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–27.

Gina-Anne Levow. 2006. The third international chinese language processing bakeoff: Word segmentation and named entity recognition. In *Proceedings of the Fifth Workshop on Chinese Language Processing, SIGHAN@COLING/ACL 2006, Sydney, Australia, July 22-23, 2006*, pages 108–117.

Bill Yuchen Lin and Wei Lu. 2018. Neural adaptation layers for cross-domain named entity recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2012–2022.

Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *International Conference on Learning Representations 2017.*

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Deep multi-task learning with shared memory for text classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 118–127.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1–10.

Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1105–1116.

Diego Mollá, Menno van Zaanen, and Daniel Smith. 2006. Named entity recognition for question answering. In *Proceedings of the 2006 Australasian Language Technology Workshop (ALTW2006)*, pages 51–58.

Nanyun Peng and Mark Dredze. 2015. Named entity recognition for chinese social media with jointly trained embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 548–554.

Nanyun Peng and Mark Dredze. 2016. Improving named entity recognition for chinese social media with word segmentation representation learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016*.

Nanyun Peng and Mark Dredze. 2017. Supplementary results for named entity recognition on chinese social media with an updated dataset. Technical report.

Kuniko Saito and Masaaki Nagata. 2003. Multi-language named-entity recognition system based on hmm. In *Proceedings of the ACL 2003 Workshop on Multilingual and Mixed-language Named Entity Recognition*.

Mohammad Golam Sohrab and Makoto Miwa. 2018. Deep exhaustive model for nested named entity recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2843–2849.

Xu Sun and Hangfeng He. 2017. F-score driven max margin neural network for named entity recognition in chinese social media. In *Proceedings of the 15th*

Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, pages 713–718.

Paola Virga and Sanjeev Khudanpur. 2003. Transliteration of proper names in cross-lingual information retrieval. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Mengqiu Wang, Wanxiang Che, and Christopher D. Manning. 2013. Effective bilingual constraints for semi-supervised learning of named entity recognizers. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA*.

Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Ann Taylor Nianwen Xue, Jeff Kaufman, Michelle Franchini, Mohammed El-Bachouti, Robert Belvin, and Ann Houston. 2011. Ontonotes release 4.0 ldc2011t03. dvd. In *Linguistic Data Consortium*.

Vikas Yadav and Steven Bethard. 2018. A survey on recent advances in named entity recognition from deep learning models. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2145–2158. Association for Computational Linguistics.

Jie Yang, Zhiyang Teng, Meishan Zhang, and Yue Zhang. 2016. Combining discrete and neural features for sequence labeling. In *Computational Linguistics and Intelligent Text Processing - 17th International Conference, CICLing 2016, Konya, Turkey, April 3-9, 2016, Revised Selected Papers, Part I*, pages 140–154.

Jie Yang and Yue Zhang. 2018. Ncrf++: An open-source neural sequence labeling toolkit. In *Proceedings of ACL 2018, System Demonstrations*, pages 74–79. Association for Computational Linguistics.

Jie Yang, Yue Zhang, and Fei Dong. 2017. Neural word segmentation with rich pretraining. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 839–849.

Jie Yang, Yue Zhang, and Shuailong Liang. 2018. Subword encoding in lattice LSTM for chinese word segmentation. *Computing Research Repository*, abs/1810.12594.

Qi Zhang, Xiaoyu Liu, and Jinlan Fu. 2018. Neural networks incorporating dictionaries for chinese word segmentation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18)*, pages 5682–5689. Springer Berlin Heidelberg.

Suxiang Zhang, Ying Qin, Juan Wen, and Xiaojie Wang. 2006. Word segmentation and named entity recognition for SIGHAN bakeoff3. In *Proceedings of the Fifth Workshop on Chinese Language Processing, SIGHAN@COLING/ACL 2006, Sydney, Australia, July 22-23, 2006*, pages 158–161.

Yue Zhang and Jie Yang. 2018. Chinese ner using lattice lstm. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018*, pages 1554–1564. Springer Berlin Heidelberg.

Junsheng Zhou, Weiguang Qu, and Fen Zhang. 2013. Chinese named entity recognition via joint identification and categorization. *Chinese journal of electronics*, 22(2):225–230.