CNN-Based Chinese NER with Lexicon Rethinking

Tao Gui[1*], **Ruotian Ma**[1*], **Qi Zhang**[1], **Lujun Zhao**[1], **Yu-Gang Jiang**[1,2] and **Xuanjing Huang**[1]
[1]School of Computer Science, Fudan University, Shanghai, China
[2]Jilian Technology Group (Video++), Shanghai, China
{tgui16, rtma15, qz, ljzhao16, ygj, xjhuang}@fudan.edu.cn

Abstract

Character-level Chinese named entity recognition (NER) that applies long short-term memory (LSTM) to incorporate lexicons has achieved great success. However, this method fails to fully exploit GPU parallelism and candidate lexicons can conflict. In this work, we propose a faster alternative to Chinese NER: a convolutional neural network (CNN)-based method that incorporates lexicons using a rethinking mechanism. The proposed method can model all the characters and potential words that match the sentence in parallel. In addition, the rethinking mechanism can address the word conflict by feeding back the high-level features to refine the networks. Experimental results on four datasets show that the proposed method can achieve better performance than both word-level and character-level baseline methods. In addition, the proposed method performs up to 3.21 times faster than state-of-the-art methods, while realizing better performance.

1 Introduction

The task of named entity recognition (NER) involves the determination of entity boundaries and the recognition of categories of named entities, which is a fundamental task in the field of natural language processing (NLP). NER plays an important role in many downstream NLP tasks, including information retrieval [Chen *et al.*, 2015], relation extraction [Bunescu and Mooney, 2005], question answering systems [Diefenbach *et al.*, 2018], and other applications. Compared with English NER, Chinese named entities are more difficult to identify due to their uncertain boundaries, complex compositions, and NE definitions within the nest [Duan and Zheng, 2011]. Hence, one intuitive way to perform Chinese NER is to first perform word segmentation and then apply word sequence labeling [Yang *et al.*, 2016; He and Sun, 2017].

However, gold-standard segmentation is rarely available in NER datasets, and word segmentation errors negatively impact the identification of named entities [Peng and Dredze, 2015; He and Sun, 2016]. The development of ways to use lexicon features to better leverage word information for NER has attracted research attention [Passos *et al.*, 2014; Zhang and Yang, 2018]. In particular, to exploit explicit word information, Zhang and Yang [2018] introduced a variant of LSTM (lattice-structured LSTM) that encodes all potential words that match a sentence. Because of its rich lexicon information, the lattice LSTM model has achieved state-of-the-art results on various datasets.

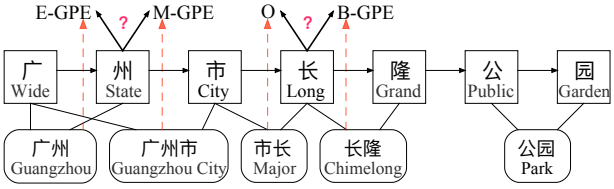

Figure 1: Example of word character lattice. RNN-based methods must recurrently encode each character with potential words in the lexicon, which reduces their speed. In addition, they may suffer from conflicts between potential words.

Although previous work using RNNs to incorporate lexicons has achieved great success, these methods continue to have difficulty with two issues. **First**, RNN-based methods fail to fully exploit GPU parallelism because of the recurrent structure, which limits their computational efficiency [Strubell *et al.*, 2017]. Specifically, lattice LSTM [Zhang and Yang, 2018] employs double recurrent transition computation across the length of the input, one for each character in a sentence and the other for matched potential words in the lexicon. As such, their speeds are limited. **Second**, they have difficulty dealing with conflicts between potential words being incorporated in the lexicon: one character may correspond to couples of potential words in the lexicon, and this conflict can misguide the model such that it predicts different labels, as shown in Figure 1. Because of the nature of the sequential processing in RNNs, it is difficult to determine which word is correct based only on the previous inputs. For example, given part of the sentence "广州市长隆(Guangzhou Chimelong)," an RNN-based model would be uncertain whether the character "长" is part of the word "市长(Major)" or "长隆(Chimelong)." The matching of the words "市长(Major)" and "长隆(Chimelong)" would guide the character "长" to be identified as 'O' and 'B-GPE,'

*Equal contributions

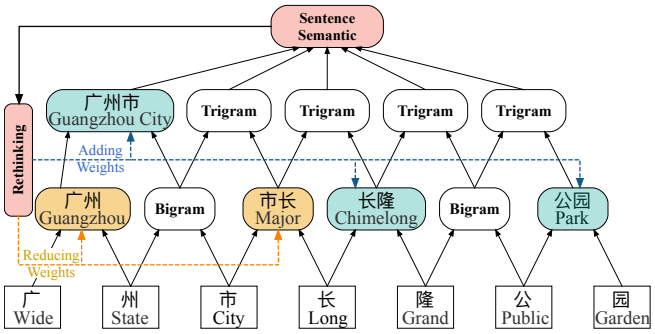

Figure 2: Schematic of CNN model incorporating lexicons with a rethinking mechanism.

respectively. More seriously, this conflict among words is ubiquitous in Chinese NER, and these ambiguities cannot be settled without reference to the whole-sentence context and high-level information [Ma *et al.*, 2014].

In this paper, we present a novel convolutional neural network with a rethinking mechanism. The **first issue** is handled using a CNN to process the whole sentence as well as all the potential words in parallel. The intuition is that when the window size of the convolution operation is set to 2, then all the potential words can easily fuse into corresponding positions [Kim, 2014]. As shown in Figure 2, words with certain lengths correspond to distinct positions in certain layers. In this manner, characters coupled with potential words can be processed in parallel. The **second issue** is addressed by the use of a rethinking mechanism [Li *et al.*, 2018]. Most existing Chinese NER models learn features using only a feedforward structure. They have no chance to modify conflicting lexicon information after seeing the whole sentence. By adding a feedback layer and feeding back the high-level features [Ma *et al.*, 2014], this rethinking mechanism can leverage the high-level semantic to refine the weights of embedded words and address the conflicts between potential words. Experimental results on four NER datasets show that the proposed method can achieve better performance than other baseline methods.

In summary, our contributions are three-fold: 1) We propose a novel CNN structure for incorporating lexicons into Chinese NER, which effectively accelerates the model training by its use of parallelism; 2) we apply a rethinking mechanism to tackle the conflict between potential words in the lexicon, and the proposed model can leverage the high-level semantic to identify the correct words; 3) experimental results on four datasets demonstrate that the proposed model can achieve better performance, and performs up to 3.21 times faster than state-of-the-art methods.

2 Related Work

Our work is inspired by three lines of research: enhancing character-level Chinese NER through lexicon information, improving computational efficiency by parallelism, and refining networks using a rethinking mechanism.

Chinese NER with lexicon. To date, using neural networks for NER has been the dominant approach [Ma and Hovy, 2016]. For Chinese NER, there have been explicit discussions comparing word-based and character-based methods, which have shown that due to the limited performance of current Chinese word segmentation, character-based name taggers can outperform their word-based counterparts [Zhang and Yang, 2018]. Zhang and Yang [2018] exploit an RNN-based lattice structure to simultaneously model characters and corresponding words in a lexicon, which prevents segmentation errors. However, lattice LSTM suffers from inefficiency and word conflict. Our approach extends these ideas from the parallel modeling of characters with potential words and a rethinking mechanism. Hence, our method is more efficient and effective.

Improving NER computational efficiency. In general, end-to-end CNNs in NLP have mainly been used for text classification [Kim, 2014]. For sequence labeling tasks, CNNs have been mainly used for low-level feature extraction[Ma and Hovy, 2016] as input for alternative architectures. Recently, Strubell *et al.* [2017] proposed iterated dilated convolutional neural networks (ID-CNNs) for sequence labeling with parallel computation. However, ID-CNNs applied a dilated window that skips over every dilation, which makes adding lexicon information impractical. In this work, we propose a novel CNN architecture that can process all the characters and lexicon information in parallel.

Refining networks with a rethinking mechanism. Previous attempts to use a rethinking mechanism in neural networks have been made in image classification to tackle issues of occlusion and noise [Li *et al.*, 2018]. Li *et al.* [2018] used the output posterior possibilities of a CNN to refine its intermediate feature maps. We extend these concepts from refining networks to tackle conflict among words.

3 Lexicon Rethinking CNNs

This paper presents a novel lexicon rethinking convolutional neural network (LR-CNN). The proposed model can process all of the characters in a sentence, as well as all of the potential words that match the sentence in parallel. By adding a feedback layer and feeding back the top-layer features, the rethinking mechanism can leverage the high-level semantic to refine the weights of embedded words and tackle the conflicts between potential words.

3.1 Lexicon-Based CNNs

CNNs have been shown to be effective for Chinese NER [Strubell *et al.*, 2017]. We propose the use of CNNs to encode the sentences and lexicons in parallel. We denote the input sentence as $\mathbf{C} = \{\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_M\}$, with character vocabulary $\mathcal{V}$, where $\mathbf{c}_m \in \mathbb{R}^d$ is the m-th character embedding. Then $\mathbf{C} \in \mathbb{R}^{d \times M}$ denotes the sentence matrix. The potential words in the lexicon that match a sequence of characters can be formulated as $\mathbf{w}_m^l = \{\mathbf{c}_m, \cdots, \mathbf{c}_{m+l-1}\}$; as such, the first letter of the word is c_m and the length of the word is l.

Take the sentence in Figure 2 as an example. The sentence consists of five potential words (e.g., $\mathbf{w}_1^2$ =广州, $\mathbf{w}_1^3$ =广州市, $\mathbf{w}_3^2$ =市长). Given the sentence $\mathbf{C}$ and potential matching words $\mathbf{w}$, we adopt character-level CNNs to encode the

character features and attention modules to incorporate the lexicon features. The CNNs apply multiple filters $\mathbf{H} \in \mathbb{R}^{d\times 2}$ with a window size of 2 to obtain the bigram information and stack the multiple layers to obtain the multigram information, as follows:

$$\begin{aligned}\mathbf{C}_m^2 &= \tanh(\langle \mathbf{C}^1[*, m:m+1], \mathbf{H}_1\rangle + b_1)\\ \mathbf{C}_m^l &= \tanh(\langle \mathbf{C}^{l-1}[*, m:m+1], \mathbf{H}_{l-1}\rangle + b_{l-1}),\end{aligned} \tag{1}$$

where $\mathbf{C}[*, m:m+1]$ is the m-to-$(m+1)$-th column in $\mathbf{C}$ and $\langle A, B\rangle = \mathrm{Tr}(AB^T)$ is the Frobenius inner product. We regard the character embedding layer as the first layer $\mathbf{C}_1$. Similarly, the feature in the l-th layer $\mathbf{C}_m^l$ represents the l gram feature.

From the perspective of the combination of characters, the l gram feature $\mathbf{C}_m^l$ corresponds to the word $\mathbf{w}_m^l$ (Figure 2). To incorporate the lexicon feature effectively, we use the vector-based attention [Chen *et al.*, 2018] to combine the l gram feature with the word feature, as follows:

$$\begin{aligned}\mathbf{i}_1 &= \sigma(W_i\mathbf{C}_m^l + U_i\mathbf{w}_m^l + b_i)\\ \mathbf{f}_1 &= \sigma(W_f\mathbf{C}_m^l + U_f\mathbf{w}_m^l + b_f)\\ \mathbf{u}_1 &= \tanh(W_u\mathbf{C}_m^l + U_u\mathbf{w}_m^l + b_u)\\ \mathbf{i}_1', \mathbf{f}_1' &= \mathrm{softmax}(\mathbf{i}_1, \mathbf{f}_1)\\ X_m^l &= \mathbf{i}_1' \odot \mathbf{u}_1 + \mathbf{f}_1' \odot \mathbf{C}_m^l,\end{aligned} \tag{2}$$

where W, U and b are the parameters of the attention module. $\odot$ refers to the element wise production, and σ represents a sigmoid operation. In this way, the model can process the characters and potential words in parallel.

3.2 Refining Networks with Lexicon Rethinking

Although using an attention module is effective to incorporate lexicons, it still has difficulty dealing with the conflicts between the potential words. The model incorporating the potential words in the lower layers cannot refer to the words in the higher layers, due to the hierarchical structure of CNN. Hence, the effective use of high-level features to tackle the ambiguity among the potential words is still a key issue [Ma *et al.*, 2014].

In this work, we treat the features at the top layer of CNN, X_m^L, as the high-level features. We then use these features to readjust the weights of the lexicon attention module by adding a feedback layer to each CNN layer, as follows:

$$\begin{aligned}\mathbf{i}_2 &= \sigma(W_i^*\mathbf{C}_m^l + U_i^*\mathbf{w}_m^l + V_iX_m^L + b_i^*)\\ \mathbf{f}_2 &= \sigma(W_f^*\mathbf{C}_m^l + U_f^*\mathbf{w}_m^l + V_fX_m^L + b_f^*)\\ \mathbf{r}_2 &= \sigma(W_r\mathbf{C}_m^l + U_r\mathbf{w}_m^l + V_rX_m^L + b_r)\\ \mathbf{u}_2 &= \tanh(W_u^*\mathbf{C}_m^l + U_u^*\mathbf{w}_m^l + b_u^*)\\ \mathbf{i}_2', \mathbf{f}_2', \mathbf{r}_2' &= \mathrm{softmax}(\mathbf{i}_2, \mathbf{f}_2, \mathbf{r}_2)\\ X_m^{l'} &= \mathbf{i}_2' \odot \mathbf{u}_2 + \mathbf{f}_2' \odot \mathbf{C}_m^l + \mathbf{r}_2' \odot X_m^L,\end{aligned} \tag{3}$$

where W, U, V and b are the parameters of the rethinking module. To avoid overfitting, W^*, U^* and b^* are the reused parameters of the attention module included to reduce the number of parameters. With high-level features participating in the reweighting of the word vectors and CNN features, the model can learn to address the conflicts between the potential words.

Consider the case in Figure 2, where the word vector "市长(Major)" is a misleading input that can give rise to an incorrect label decision. Without the features of "广州市(Guangzhou City)" and "长隆(Chimelong)" in the higher layer, the model may focus more on the misleading word vector. However, the rethink vector from the upstream layer can decrease the weight of the word vector "市长(Major)" in the output features, so as to correct the label decision.

Through the CNN encoder and lexicon attention module, the model obtains the respective l gram features X_m^l within each layer of the model. These feature values correspond to the multi-scale l-grams, of which some are redundant (i.e., unigram X_m^1, bigram X_m^2, ..., and L-gram X_m^L). In this work, we propose the use of the multi-scale feature attention [Wang *et al.*, 2018] to adaptively selects the features of different scales in each position of a sentence, as follows:

$$\begin{aligned}s_m^l &= \sum_{d=1}^{D} X_m^{l'}[d], \quad \alpha_m^l = \frac{\exp(s_m^l)}{\sum_{l=1}^{L}\exp(s_m^l)}\\ X_m^{att} &= \sum_{l=1}^{L}\alpha_m^l X_m^{l'}.\end{aligned} \tag{4}$$

After being processed by the multi-scale attention module, the final representation $\mathbf{X}^{att} = [X_1^{att}, X_2^{att}, \cdots, X_M^{att}]$ would be fed into the CRF layer for the prediction.

3.3 Predicting NER with CRF

Suppose that $\mathbf{y} = \{y_1, \cdots, y_M\}$ represents a sequence of labels for sentence $\mathbf{C}$, and $\mathcal{Y}(\mathbf{C})$ denotes the set of possible label sequences for $\mathbf{C}$. The probability of the label sequence $\mathbf{y}$ is:

$$p(\mathbf{y}|\mathbf{X}^{att}; W, b) = \frac{\prod_{m=1}^{M}\psi_m(y_{m-1}, y_m, \mathbf{X}^{att})}{\sum_{y'\in\mathcal{Y}(\mathbf{C})}\prod_{m=1}^{M}\psi_m(y'_{m-1}, y'_m, \mathbf{X}^{att})}, \tag{5}$$

where $\psi_i(y_{m-1}, y_m, \mathbf{X}^{att}) = \exp(W_m\mathbf{X}^{att} + b_m)$ is the potential function and W_m and b_m are the weight vector and bias, corresponding to label pair (y_{m-1}, y_m), respectively.

During the training, we optimize the parameters of the model to maximize the following conditional likelihood:

$$\mathcal{L}(\mathbf{W}, \mathbf{b}) = \sum \log p(\mathbf{y}|\mathbf{C}; \mathbf{W}, \mathbf{b}), \tag{6}$$

where $\mathbf{y}$ is the true labels of sentence $\mathbf{C}$, and $\mathbf{W}, \mathbf{b}$ are the parameters of the model. During the decoding process, we search for the label sequence $\mathbf{y}^*$ with the highest conditional probability:

$$\mathbf{y}^* = \underset{\mathbf{y}\in\mathcal{Y}(\mathbf{C})}{\mathrm{argmax}}\, p(\mathbf{y}|\mathbf{C}; \mathbf{W}, \mathbf{b}) \tag{7}$$

4 Experimental Setup

In this section, we describe the datasets that include the newswire and social media domains. We then detail the baseline methods applied, including the word- and character-based neural Chinese NER under different settings. Finally, we detail the configuration of the proposed model.

Datasets	Type	Train	Dev	Test
OntoNotes	Sentence	15.7k	4.3k	4.3k
	Char	491.9k	200.5k	208.1k
MSRA	Sentence	46.4k	-	4.4k
	Char	2169.9k	-	172.6k
Weibo	Sentence	1.4k	0.27k	0.27k
	Char	73.8k	14.5	14.8k
Resume	Sentence	3.8k	0.46	0.48k
	Char	124.1k	13.9k	15.1k

Table 1: Statistics of datasets.

4.1 Datasets

We evaluate the proposed method on four datasets, including OntoNotes [Weischedel *et al.*, 2011], MSRA [Levow, 2006], Weibo NER [Peng and Dredze, 2015; He and Sun, 2016], and Resume NER [Zhang and Yang, 2018]. The splitting methods follow those in [Zhang and Yang, 2018].

The OntoNotes and MSRA are the newswire domain, where gold-standard segmentation is available in the training data. For OntoNotes, gold segmentation is also available for the development and test data; however, no segmentation is available for the MSRA test data. The Weibo and Resume NER come from social media. There is no segmentation in the Weibo and Resume datasets. A description of the datasets is presented in Table 1.

We use the pretrained character embeddings and lexicon embeddings[1] trained using word2vec [Mikolov *et al.*, 2013], over the automatically segmented Chinese GigaWord[2]. The lexicon consists of 704.4k words, containing 5.7k single-character, 291.5k two-character, 278.1k three-character words, and 129.1k other words.

4.2 Comparison Methods

As baselines for comparison, we applied several classic and state-of-the-art methods on the four datasets. In addition, to verify the effectiveness of the proposed method, we compare the word- and character-level methods that utilize the bichar, softword, and lexicon features.

For the dataset without gold segmentation, we first train the open source segmentation toolkit[3] on the training data. We then use it to automatically segment the datasets. Finally, we apply the word-level NER methods on these segmented datasets to evaluate their performances, which are denoted as **Auto seg**. For the dataset with gold segmentation, we directly apply the word-level NER methods, which are denoted as **Gold seg**. We also evaluate the character-level methods without using segmentation information, which are denoted as **No seg**. All the methods evaluated are as follows:

LSTM. A bi-directional LSTM [Hochreiter and Schmidhuber, 1997] is used to obtain a left-to-right hidden state $\overrightarrow{h}_i^w$ and a right-to-left hidden state $\overleftarrow{h}_i^w$, which are concatenated for the NER prediction.

[1]https:// github.com/jiesutd/LatticeLSTM.

[2]https://catalog.ldc.upenn.edu/ LDC2011T13

[3]https://github.com/lancopku/PKUSeg-python

CNN. We apply a standard CNN [Kim, 2014] structure on the character or word sequence to obtain its multiple gram representation for the NER prediction.

Word-level model + char + bichar. To extract the morphological and combination information from the words, we first concatenate the character embedding and bigram embedding to represent the character $\mathbf{x}_m^c$. We then use a bi-directional LSTM on the character sequence to obtain the character-level features. Finally, we augment the word-level model by combining word embedding with the character-level features as: $\mathbf{x}_m^w = [\mathbf{w}_m \oplus \mathbf{x}_m^c]$.

Character-level model + bichar + softword. We use the BMES scheme for representing the segmentation. In addition, we use the same bigram feature $\mathbf{b}_m$ as a word-level model. Then, the character feature can be represented as: $\mathbf{x}_m^c = [\mathbf{c}_m \oplus \mathbf{b}_m \oplus \mathbf{seg}(c_m)]$.

Dilated CNN. Dilated CNN [Strubell *et al.*, 2017] stacks several of the CNN layers that applied a dilated window skips over every dilation. This method achieves great performance in the English NER task.

Lattice LSTM. Lattice LSTM [Zhang and Yang, 2018] can model the characters in sequence and explicitly leverages word information through Gated recurrent cells, which can avoid segmentation errors. The lattice LSTM achieved state-of-the-art performance on the four datasets.

4.3 Hyper-Parameter Settings

For all four of the datasets, we used the Adamax [Kingma and Ba, 2014] optimization to train our networks. The initial learning rate was set at 0.0015, with a decay rate of 0.05. To avoid overfitting, we employed the dropout technique (50% dropout rate) on the character embeddings, lexicon embeddings and each layer of the CNNs. The character embeddings and lexicon embeddings were initialized by a pre-trained embedding and then fine-tuned during the training. The character embedding size and lexicon embedding size were set to 50. For the biggest dataset, MSRA, we used five layers of CNNs with an output channel size of 300. For the other datasets, we used four layers of CNNs with an output channel size of 128. We used early stopping, based on the performance on the development set. Our code are released at https://github.com/guitaowufeng/LR-CNN.

5 Results and Analysis

In this section, we detail the performance of the proposed baseline models. We present the results of a series of experiments to demonstrate the effectiveness of the lexicon and rethinking mechanism.

5.1 Method Comparison

For OntoNotes, gold-standard segmentation is available in the entire dataset. No segmentation is available for the MSRA test sections, nor is the Weibo and Resume datasets. As a result, we study the oracle situations where gold segmentation is given on OntoNotes and evaluate the automatic/no segmentation situations on the other three datasets.

input	Models	P	R	F1
Gold seg	Yang *et al.* 2016	65.59	71.84	68.57
	Yang *et al.* 2016*†	72.98	**80.15**	**76.40**
	Che *et al.* 2013*	77.71	72.51	75.02
	Wang *et al.* 2013*	76.43	72.32	74.32
	Word-level LSTM	76.66	63.60	69.52
	+ char+bichar	**78.62**	73.13	75.77
	Word-level CNN	66.84	62.99	64.86
	+ char+bichar	68.22	72.37	70.24
	Word-level Dilated CNN	76.90	63.42	69.51
Auto seg	Word-level LSTM	72.84	59.72	65.63
	+ char+bichar	73.36	**70.12**	**71.70**
	Word-level CNN	54.62	55.20	54.91
	+ char+bichar	64.69	65.09	64.89
	Word-level Dilated CNN	**74.60**	56.31	64.18
No seg	Char-level LSTM	68.79	60.35	64.30
	+ bichar+ softword	74.36	69.43	71.89
	Char-level CNN	56.78	60.99	58.81
	+ bichar + softword	59.60	65.14	62.25
	Char-level Dilated CNN	59.42	57.43	58.41
	Lattice LSTM	76.35	71.56	73.88
	LR-CNN	**76.40**	**72.60**	**74.45**

Table 2: Main results on OntoNotes.

Models	P	R	F1
Chen *et al.* 2006	91.22	81.71	86.20
Zhang *et al.* 2006*	92.20	90.18	91.18
Zhou *et al.* 2013	91.86	88.75	90.28
Lu *et al.* 2016	-	-	87.94
Dong *et al.* 2016	91.28	90.62	90.95
Lattice LSTM	93.57	92.79	93.18
LR-CNN	**94.50**	**92.93**	**93.71**

Table 3: Main results on MSRA.

Table 2[4] illustrates a variety of settings for the word- and character-based Chinese NER. In the gold or automatic segmentation settings, the char and bichar features can enrich the word representations in the word-level models to obtain a better performance than those without char and bichar features. In particular, these models can achieve results that are competitive to the state-of-the-art [Che *et al.*, 2013; Wang *et al.*, 2013]. However, due to the influence of the word segmentation errors, the models on the automatic segmentation dataset would perform worse than those on the gold segmentation dataset. In addition, the gold segmentation is not available in most datasets. Hence, the previous work explores the character-based methods to avoid the need for word segmentation. The LR-CNN outperforms character-level lattice LSTM by 0.57% in F1 score. It also results in an almost 3 percent improvement over the LSTM with bichar and softword features. In addition, because the processing of the characters and lexicons are conducted in parallel, LR-CNN has an efficiency advantage (Table 6).

Tables 3, 4, and 5 present the comparisons between the classic and state-of-the-art methods on the MSRA, Weibo, and Resume datasets. The existing classic methods explore the rich handcrafted features [Chen *et al.*, 2006; Zhang *et al.*, 2006; Zhou *et al.*, 2013], character embedding features [Lu *et al.*, 2016], segmentation embedding features [Peng and Dredze, 2016], and radical embedding features [Dong *et al.*, 2016]. He and Sun 2017 leverage the cross-domain and semi-supervised data for the Chinese NER. Our LR-CNN model significantly outperforms these models. In addition, with a better manner to incorporate the lexicons, LR-CNN also outperforms the strong baseline lattice LSTM on all of the datasets.

[4]In Table 2, 3 and 4, the results with $*$ refer to the model with external labeled data for semi-supervised learning. those with † mean that the model also uses discrete features.

Models	NE	NM	Overall
Peng and Dredze 2015	51.96	61.05	56.05
Peng and Dredze 2016*	55.28	62.97	58.99
He and Sun 2016	50.60	59.32	54.82
He and Sun 2017*	54.50	62.17	58.23
Lattice LSTM	53.04	62.25	58.79
LR-CNN	**57.14**	**66.67**	**59.92**

Table 4: Main results on Weibo NER. NE, NM and Overall denote F1-scores for named entities, nominal entities (excluding named entities) and both, respectively.

Models	P	R	F1
Word-level LSTM	93.72	93.44	93.58
+ char+ bichar	94.07	94.42	94.24
Char-level LSTM	93.66	93.31	93.48
+ bichar+ softword	94.53	94.29	94.41
Char-level CNN	91.58	94.05	92.80
+ bichar + softword	92.69	94.85	93.75
Lattice LSTM	94.81	94.11	94.46
LR-CNN	**95.37**	**94.84**	**95.11**

Table 5: Main results on Resume NER.

5.2 Efficiency Advantage

Our LR-CNN not only achieves better F1 score results than the baseline models, but it is also faster. Table 6 lists the relative decoding time on all four of the development sets, as compared to the lattice LSTM. We report the decoding time using the same batch size for each method.

Without the CRF, the LR-CNN decodes up to 3.21 times faster than the lattice LSTM. With Viterbi decoding, the gap closes somewhat, but the LR-CNN still has better efficiency (i.e., about an average of 1.91 times faster than the lattice LSTM). The LR-CNN with CRF is even faster than the lattice LSTM without CRF. In addition, we study the degree of the decline in model performance when the models get rid of the CRF layer. We find that the F1 score of the LR-CNN decreases, on average, by 5.75% on the four datasets. In contrast, the lattice LSTM decreases by 6.10%, showing that the LR-CNN is a better structure for the token encoder.

5.3 Influence of Sentence Length

To investigate the influence of the different sentence lengths, we analyze the performance of the LR-CNN and lattice LSTM on the OntoNotes dataset. The dataset is split into

Models		OntoNotes		MSRA		Weibo		Resume	
		F1	Speedup	F1	Speedup	F1	Speedup	F1	Speedup
W/ CRF	Lattice LSTM	73.88	1×	93.18	1×	58.79	1×	94.46	1×
	LR-CNN	74.45	2.23×	93.71	1.57×	59.92	2.41×	95.11	1.44×
W/O CRF	Lattice LSTM	64.74	1.14×	85.38	1.12×	52.38	1.21×	93.43	1.24×
	LR-CNN	69.48	2.94×	84.14	1.95×	53.33	3.21×	93.94	2.43×

Table 6: Relative test-time speed of different models with CRF (w/ CRF) and without CRF (w/o CRF).

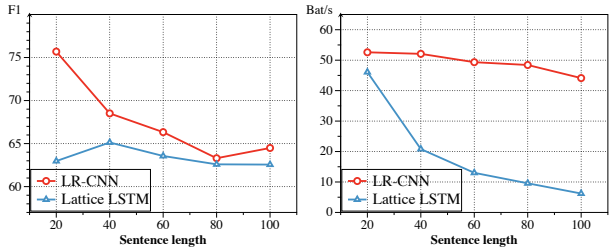

Figure 3: F1 and speed against sentence length. Bat/s refers to the number of batches can be processed per second.

Models	OntoNotes	MSRA	Weibo	Resume
LR-CNN	74.45	93.71	59.92	95.11
- Rethink	73.09	93.58	57.86	95.04
- Lexicon	65.31	86.81	49.58	94.27
- Lexicon - Rethink	59.86	87.81	50.75	94.47

Table 7: An ablation study of the proposed model.

five parts according to the sentence length. To rule out the impact of the CRF, we evaluate the models by removing the CRF layer. Figure 3 demonstrates the F1 score and processing speed on the different sentence lengths.

The results reveal that the LR-CNN outperforms the lattice LSTM on all of the datasets with different sentence lengths. In particular, the F1 score for LR-CNN is nearly 12% greater than that of lattice LSTM when the sentence length is less than 20 characters. Hence, LR-CNN has a much greater advantage than the lattice LSTM in dealing with short sentences. In addition, we evaluate the processing speed of the sentences with different lengths. We find that with the sentence length increases, the speed of the LR-CNN is relatively stable, while the speed of the lattice LSTM decreases substantially. In particular, when processing the sentence of which the length is greater than 100, LR-CNN is 5 times faster than the lattice LSTM.

5.4 Ablation Study

To study the contribution of the main components in the LR-CNN, we ran an ablation study on all four datasets. The results are reported in Table 7. Overall, we find that:

(1) Lexicons play an important role in character-level Chinese NER. Compared with models incorporating lexicons, the performance of the model without lexicons is seriously degraded, and the F1 score decreases by 9% on the Weibo dataset.

(2) Rethinking mechanisms can further improve the performance of the models with lexicons, because the rethinking mechanisms can tackle the conflicts between the potential words that match the same characters. Table 8 illustrates two examples where the rethinking mechanism successfully tackles the conflict words and makes correct predictions.

Sentence	会议八月九日在汕头大学举行 The meeting was held at Shantou University on August 9th.
Gold seg	会议 八月 九日 在 汕头大学 举行 The meeting, August, 9th, was, Shantou University, held
Lattice LSTM	O O O O O O O O O O O O O O O 会议八月九日在汕头大学举行
LR-CNN w/o rethink	O O O O O O O B E B E O O (GPE) 会议八月九日在 汕头 大学 举行
LR-CNN	O O O O O O O B M M E O O (GPE) 会议八月九日在 汕头大学 举行
Sentence	中国内陆大省四川 TSichuan, China's inland province.
Gold seg	中国 内陆 大省 四川 China, inland, province, TSichuan
Lattice LSTM	B M M M M M M E (ORG) 中国内陆大省四川
LR-CNN w/o rethink	B E O B M E B E (GPE) 中国 内 陆大省 四川
LR-CNN	B E O O O O B E (GPE) 中国 内陆 大省 四川

Table 8: Examples of OntoNotes dataset. Characters with blue and red highlights represent incorrect and correct entities, respectively.

(3) The rethinking mechanism may benefit models without lexicons. Although the lexicons are not available, the LR-CNN - Lexicon is 5% better than itself without the rethinking mechanism in the F1 score on OntoNotes dataset.

6 Conclusion

In this work, we propose a novel convolutional neural network to incorporate lexicons with a rethinking mechanism. This network can model all the characters coupled with the potential words that match the sentence in parallel. To solve the conflicts between the potential words, the proposed model can refine the networks by adding a feedback layer to feed back the high-level features. We evaluate the proposed model on four Chinese NER datasets. The experimental results illustrate that the proposed method can significantly improve the performance when compared with that of the other baseline approaches. In addition, the proposed method is up to 3.21 times faster than the state-of-the-art methods.

Acknowledgements

The authors wish to thank the anonymous reviewers for their helpful comments. This work was partially funded by China National Key R&D Program (No. 2018YFC0831105, 2017YFB1002104, 2018YFB1005104), National Natural Science Foundation of China (No. 61751201, 61532011), Shanghai Municipal Science and Technology Major Project (No.2018SHZDZX01), STCSM (No.16JC1420401, 17JC1420200), ZJLab.

References

[Bunescu and Mooney, 2005] Razvan C Bunescu and Raymond J Mooney. A shortest path dependency kernel for relation extraction. In *HLT—EMNLP*, 2005.

[Che *et al.*, 2013] Wanxiang Che, Mengqiu Wang, Christopher D Manning, and Ting Liu. Named entity recognition with bilingual constraints. In *NAACL*, pages 52–62, 2013.

[Chen *et al.*, 2006] Aitao Chen, Fuchun Peng, Roy Shan, and Gordon Sun. Chinese named entity recognition with conditional probabilistic models. In *SIGHAN Workshop on Chinese Language Processing*, 2006.

[Chen *et al.*, 2015] Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. Event extraction via dynamic multi-pooling convolutional neural networks. In *ACL—IJCNLP*, volume 1, pages 167–176, 2015.

[Chen *et al.*, 2018] Qian Chen, Zhen-Hua Ling, and Xiaodan Zhu. Enhancing sentence embedding with generalized pooling. In *COLING*, pages 1815–1826, 2018.

[Diefenbach *et al.*, 2018] Dennis Diefenbach, Vanessa Lopez, Kamal Singh, and Pierre Maret. Core techniques of question answering systems over knowledge bases: a survey. *KAIS*, 55(3):529–569, 2018.

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

[Duan and Zheng, 2011] Huanzhong Duan and Yan Zheng. A study on features of the crfs-based chinese named entity recognition. *IJAI*, 3(2):287–294, 2011.

[He and Sun, 2016] Hangfeng He and Xu Sun. F-score driven max margin neural network for named entity recognition in chinese social media. *arXiv preprint arXiv:1611.04234*, 2016.

[He and Sun, 2017] Hangfeng He and Xu Sun. A unified model for cross-domain and semi-supervised named entity recognition in chinese social media. In *AAAI*, 2017.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[Kim, 2014] Yoon Kim. Convolutional neural networks for sentence classification. *EMNLP*, pages 1746–1751, 2014.

[Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Levow, 2006] Gina-Anne Levow. The third international chinese language processing bakeoff: Word segmentation and named entity recognition. In *SIGHAN Workshop on Chinese Language Processing*, pages 108–117, 2006.

[Li *et al.*, 2018] Xin Li, Zequn Jie, Jiashi Feng, Changsong Liu, and Shuicheng Yan. Learning with rethinking: Recurrently improving convolutional neural networks through feedback. *Pattern Recognition*, 2018.

[Lu *et al.*, 2016] Yanan Lu, Yue Zhang, and Dong-Hong Ji. Multi-prototype chinese character embedding. In *LREC*, 2016.

[Ma and Hovy, 2016] Xuezhe Ma and Eduard Hovy. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *ACL*, volume 1, pages 1064–1074, 2016.

[Ma *et al.*, 2014] Guojie Ma, Xingshan Li, and Keith Rayner. Word segmentation of overlapping ambiguous strings during chinese reading. *Journal of Experimental Psychology: Human Perception and Performance*, 40(3):1046, 2014.

[Mikolov *et al.*, 2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NeurIPS*, pages 3111–3119, 2013.

[Passos *et al.*, 2014] Alexandre Passos, Vineet Kumar, and Andrew McCallum. Lexicon infused phrase embeddings for named entity resolution. *CoNLL-2014*, page 78, 2014.

[Peng and Dredze, 2015] Nanyun Peng and Mark Dredze. Named entity recognition for chinese social media with jointly trained embeddings. In *EMNLP*, 2015.

[Peng and Dredze, 2016] Nanyun Peng and Mark Dredze. Improving named entity recognition for chinese social media with word segmentation representation learning. In *ACL*, page 149, 2016.

[Strubell *et al.*, 2017] Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. Fast and accurate entity recognition with iterated dilated convolutions. In *EMNLP*, pages 2670–2680, 2017.

[Wang *et al.*, 2013] Mengqiu Wang, Wanxiang Che, and Christopher D Manning. Effective bilingual constraints for semi-supervised learning of named entity recognizers. In *AAAI*, 2013.

[Wang *et al.*, 2018] Shiyao Wang, Minlie Huang, and Zhidong Deng. Densely connected cnn with multi-scale feature attention for text classification. In *IJCAI*, 2018.

[Weischedel *et al.*, 2011] Ralph Weischedel, Sameer Pradhan, Lance Ramshaw, Martha Palmer, Nianwen Xue, Mitchell Marcus, Ann Taylor, Craig Greenberg, Eduard Hovy, Robert Belvin, et al. Ontonotes release 4.0. *LDC2011T03, Philadelphia, Penn.: Linguistic Data Consortium*, 2011.

[Yang *et al.*, 2016] Jie Yang, Zhiyang Teng, Meishan Zhang, and Yue Zhang. Combining discrete and neural features for sequence labeling. In *CICLing*. Springer, 2016.

[Zhang and Yang, 2018] Yue Zhang and Jie Yang. Chinese ner using lattice lstm. *ACL*, 1:1554–1564, 2018.

[Zhang *et al.*, 2006] Suxiang Zhang, Ying Qin, Juan Wen, and Xiaojie Wang. Word segmentation and named entity recognition for sighan bakeoff3. In *SIGHAN Workshop on Chinese Language Processing*, pages 158–161, 2006.

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