Simplify the Usage of Lexicon in Chinese NER

Ruotian Ma[1*], **Minlong Peng**[1*], **Qi Zhang**[1,3], **Zhongyu Wei**[2,3], **Xuanjing Huang**[1]
[1]Shanghai Key Laboratory of Intelligent Information Processing,
School of Computer Science, Fudan University
[2]School of Data Science, Fudan University
[3]Research Institute of Intelligent and Complex Systems, Fudan University
{rtma19,mlpeng16,qz,zywei,xjhuang}@fudan.edu.cn

Abstract

Recently, many works have tried to augment the performance of Chinese named entity recognition (NER) using word lexicons. As a representative, Lattice-LSTM (Zhang and Yang, 2018) has achieved new benchmark results on several public Chinese NER datasets. However, Lattice-LSTM has a complex model architecture. This limits its application in many industrial areas where real-time NER responses are needed.

In this work, we propose a simple but effective method for incorporating the word lexicon into the character representations. This method avoids designing a complicated sequence modeling architecture, and for any neural NER model, it requires only subtle adjustment of the character representation layer to introduce the lexicon information. Experimental studies on four benchmark Chinese NER datasets show that our method achieves an inference speed up to 6.15 times faster than those of state-of-the-art methods, along with a better performance. The experimental results also show that the proposed method can be easily incorporated with pre-trained models like BERT. [1]

1 Introduction

Named Entity Recognition (NER) is concerned with the identification of named entities, such as persons, locations, and organizations, in unstructured text. NER plays an important role in many downstream tasks, including knowledge base construction (Riedel et al., 2013), information retrieval (Chen et al., 2015), and question answering (Diefenbach et al., 2018). In languages where words are naturally separated (e.g., English), NER has been conventionally formulated as a sequence labeling problem, and the state-of-the-art results have been achieved using neural-network-based models (Huang et al., 2015; Chiu and Nichols, 2016; Liu et al., 2018).

Compared with NER in English, Chinese NER is more difficult since sentences in Chinese are not naturally segmented. Thus, a common practice for Chinese NER is to first perform word segmentation using an existing CWS system and then apply a word-level sequence labeling model to the segmented sentence (Yang et al., 2016; He and Sun, 2017b). However, it is inevitable that the CWS system will incorrectly segment query sentences. This will result in errors in the detection of entity boundary and the prediction of entity category in NER. Therefore, some approaches resort to performing Chinese NER directly at the character level, which has been empirically proven to be effective (He and Wang, 2008; Liu et al., 2010; Li et al., 2014; Liu et al., 2019; Sui et al., 2019; Gui et al., 2019b; Ding et al., 2019).

A drawback of the purely character-based NER method is that the word information is not fully exploited. With this consideration, Zhang and Yang, (2018) proposed Lattice-LSTM for incorporating word lexicons into the character-based NER model. Moreover, rather than heuristically choosing a word for the character when it matches multiple words in the lexicon, the authors proposed to preserve all words that match the character, leaving the subsequent NER model to determine which word to apply. To realize this idea, they introduced an elaborate modification to the sequence modeling layer of the LSTM-CRF model (Huang et al., 2015). Experimental studies on four Chinese NER datasets have verified the effectiveness of Lattice-LSTM.

However, the model architecture of Lattice-LSTM is quite complicated. In order to introduce lexicon information, Lattice-LSTM adds several additional edges between nonadjacent characters

*Equal contribution.

[1]The source code of this paper is publicly available at https://github.com/v-mipeng/LexiconAugmentedNER.

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 5951–5960

in the input sequence, which significantly slows its training and inference speeds. In addition, it is difficult to transfer the structure of Lattice-LSTM to other neural-network architectures (e.g., convolutional neural networks and transformers) that may be more suitable for some specific tasks.

In this work, we propose a simpler method to realize the idea of Lattice-LSTM, i.e., incorporating all the matched words for each character to a character-based NER model. The first principle of our model design is to achieve a fast inference speed. To this end, we propose to encode lexicon information in the character representations, and we design the encoding scheme to preserve as much of the lexicon matching results as possible. Compared with Lattice-LSTM, our method avoids the need for a complicated model architecture, is easier to implement, and can be quickly adapted to any appropriate neural NER model by adjusting the character representation layer. In addition, ablation studies show the superiority of our method in incorporating more complete and distinct lexicon information, as well as introducing a more effective word-weighting strategy. The contributions of this work can be summarized as follows:

- We propose a simple but effective method for incorporating word lexicons into the character representations for Chinese NER.
- The proposed method is transferable to different sequence-labeling architectures and can be easily incorporated with pre-trained models like BERT (Devlin et al., 2018).

We performed experiments on four public Chinese NER datasets. The experimental results show that when implementing the sequence modeling layer with a single-layer Bi-LSTM, our method achieves considerable improvements over the state-of-the-art methods in both inference speed and sequence labeling performance.

2 Background

In this section, we introduce several previous works that influenced our work, including the Softword technique and Lattice-LSTM.

2.1 Softword Feature

The Softword technique was originally used for incorporating word segmentation information into downstream tasks (Zhao and Kit, 2008; Peng and Dredze, 2016). It augments the character representation with the embedding of its corresponding segmentation label:

$$x_j^c \leftarrow [x_j^c; \boldsymbol{e}^{seg}(seg(c_j))]. \quad (1)$$

Here, $seg(c_j) \in \mathcal{Y}_{seg}$ denotes the segmentation label of the character c_j predicted by the word segmentor, $\boldsymbol{e}^{seg}$ denotes the segmentation label embedding lookup table, and typically $\mathcal{Y}_{seg} = \{\text{B}, \text{M}, \text{E}, \text{S}\}$.

However, gold segmentation is not provided in most datasets, and segmentation results obtained by a segmenter can be incorrect. Therefore, segmentation errors will inevitably be introduced through this approach.

2.2 Lattice-LSTM

Lattice-LSTM designs to incorporate lexicon information into the character-based neural NER model. To achieve this purpose, lexicon matching is first performed on the input sentence. If the subsequence $\{c_i, \cdots, c_j\}$ of the sentence matches a word in the lexicon for $i < j$, a directed edge is added from c_i to c_j. All lexicon matching results related to a character are preserved by allowing the character to be connected with multiple other characters. *Intrinsically, this practice converts the input form of a sentence from a chain into a graph.*

In a normal LSTM layer, the hidden state h_i and the memory cell c_i of each time step is updated by:

$$h_i, c_i = f(h_{j-1}, c_{j-1}, x_j^c), \quad (2)$$

However, in order to model the graph-based input, Lattice-LSTM introduces an elaborate modification to the normal LSTM. Specifically, let $s_{<*,j>}$ denote the list of sub-sequences of sentence s that match the lexicon and end with c_j, $h_{<*,j>}$ denote the corresponding hidden state list $\{h_i, \forall s_{<i,j>} \in s_{<*,j>}\}$, and $c_{<*,j>}$ denote the corresponding memory cell list $\{c_i, \forall s_{<i,j>} \in s_{<*,j>}\}$. In Lattice-LSTM, the hidden state h_j and memory cell c_j of c_j are now updated as follows:

$$h_j, c_j = f(h_{j-1}, c_{j-1}, x_j^c, s_{<*,j>}, h_{<*,j>}, c_{<*,j>}), \quad (3)$$

where f is a simplified representation of the function used by Lattice-LSTM to perform memory update.

From our perspective, there are two main advantages to Lattice-LSTM. First, it preserves all the possible lexicon matching results that are related to

a character, which helps avoid the error propagation problem introduced by heuristically choosing a single matching result for each character. Second, it introduces pre-trained word embeddings to the system, which greatly enhances its performance.

However, efficiency problems exist in Lattice-LSTM. Compared with normal LSTM, Lattice-LSTM needs to additionally model $s_{<*,j>}$, $h_{<*,j>}$, and $c_{<*,j>}$ for memory update, which slows the training and inference speeds. Additionally, due to the complicated implementation of f, it is difficult for Lattice-LSTM to process multiple sentences in parallel (in the published implementation of Lattice-LSTM, the batch size was set to 1). These problems limit its application in some industrial areas where real-time NER responses are needed.

3 Approach

In this work, we sought to retain the merits of Lattice-LSTM while overcoming its drawbacks. To this end, we propose a novel method in which lexicon information is introduced by simply adjusting the character representation layer of an NER model. We refer to this method as *SoftLexicon*. As shown in Figure 1, the overall architecture of the proposed method is as follows. First, each character of the input sequence is mapped into a dense vector. Next, the SoftLexicon feature is constructed and added to the representation of each character. Then, these augmented character representations are put into the sequence modeling layer and the CRF layer to obtain the final predictions.

3.1 Character Representation Layer

For a character-based Chinese NER model, the input sentence is seen as a character sequence $s = \{c_1, c_2, \cdots, c_n\} \in \mathcal{V}_c$, where $\mathcal{V}_c$ is the character vocabulary. Each character c_i is represented using a dense vector (embedding):

$$x_i^c = \boldsymbol{e}^c(c_i), \tag{4}$$

where $\boldsymbol{e}^c$ denotes the character embedding lookup table.

Char + bichar. In addition, Zhang and Yang, (2018) has proved that character bigrams are useful for representing characters, especially for those methods not using word information. Therefore, it is common to augment the character representations with bigram embeddings:

$$x_i^c = [\boldsymbol{e}^c(c_i); \boldsymbol{e}^b(c_i, c_{i+1})], \tag{5}$$

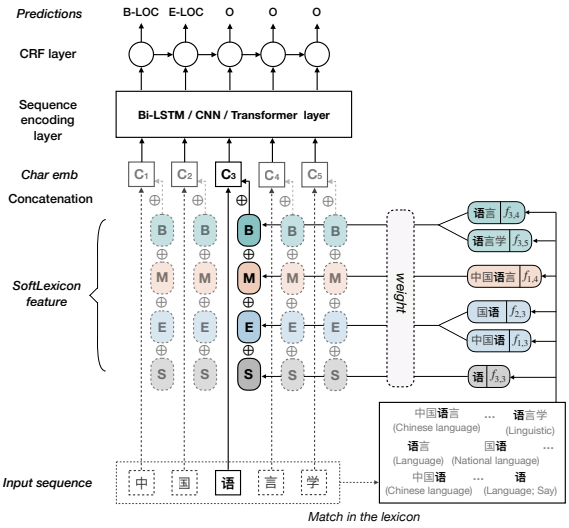

Figure 1: The overall architecture of the proposed method.

where $\boldsymbol{e}^b$ denotes the bigram embedding lookup table.

3.2 Incorporating Lexicon Information

The problem with the purely character-based NER model is that it fails to exploit word information. To address this issue, we proposed two methods, as described below, to introduce the word information into the character representations. In the following, for any input sequence $s = \{c_1, c_2, \cdots, c_n\}$, $w_{i,j}$ denotes its sub-sequence $\{c_i, c_{i+1}, \cdots, c_j\}$.

ExSoftword Feature

The first conducted method is an intuitive extension of the Softword method, called *ExSoftword*. Instead of choosing one segmentation result for each character, it proposes to retain all possible segmentation results obtained using the lexicon:

$$\boldsymbol{x}_j^c \leftarrow [\boldsymbol{x}_j^c; \boldsymbol{e}^{seg}(segs(c_j)], \tag{6}$$

where $segs(c_j)$ denotes all segmentation labels related to c_j, and $\boldsymbol{e}^{seg}(segs(c_j))$ is a 5-dimensional multi-hot vector with each dimension corresponding to an item of {B, M, E, S, O}.

As an example presented in Figure 2, the character c_7 ("西") occurs in two words, $w_{5,8}$ ("中山西路") and $w_{6,7}$ ("山西"), that match the lexicon, and it occurs in the middle of "中山西路" and the end of "山西". Therefore, its corresponding segmentation result is {M, E}, and its character representation is enriched as follows:

$$\boldsymbol{x}_7^c \leftarrow [\boldsymbol{x}_7^c; \boldsymbol{e}^{seg}(\{M, E\})]. \tag{7}$$

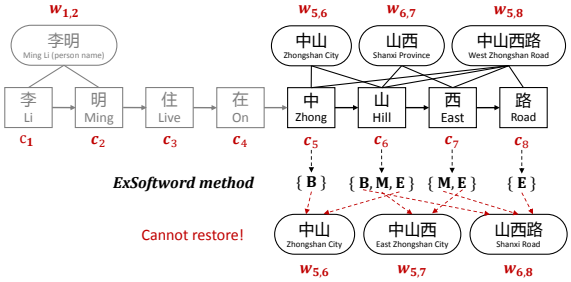

Figure 2: The ExSoftword method.

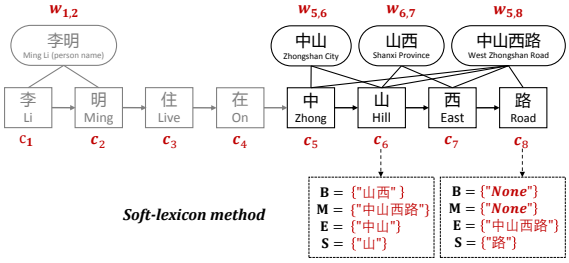

Figure 3: The SoftLexicon method.

Here, the second and third dimensions of $\boldsymbol{e}^{seg}(\cdot)$ are set to 1, and the rest dimensions are set to 0.

The problem of this approach is that it cannot fully inherit the two merits of Lattice-LSTM. First, it fails to introduce pre-trained word embeddings. Second, it still losses information of the matching results. As shown in Figure 2, the constructed ExSoftword feature for characters $\{c_5, c_6, c_7, c_8\}$ is $\{\{\mathrm{B}\}, \{\mathrm{B}, \mathrm{M}, \mathrm{E}\}, \{\mathrm{M}, \mathrm{E}\}, \{\mathrm{E}\}\}$. However, given this constructed sequence, there exists more than one corresponding matching results, such as $\{w_{5,6}$ ("中山"), $w_{5,7}$ ("中山西"), $w_{6,8}$ ("山西路")$\}$ and $\{w_{5,6}$ ("中 山"), $w_{6,7}$ ("山 西"), $w_{5,8}$ ("中 山 西 路")$\}$. Therefore, we cannot tell which is the correct result to be restored.

SoftLexicon

Based on the analysis on Exsoftword, we further developed the *SoftLexicon* method to incorporate the lexicon information. The SoftLexicon features are constructed in three steps.

Categorizing the matched words. First, to retain the segmentation information, all matched words of each character c_i is categorized into four word sets "BMES", which is marked by the four segmentation labels. For each character c_i in the input sequence $= \{c_1, c_2, \cdots, c_n\}$, the four set is constructed by:

$$\begin{aligned}
\mathrm{B}(\mathrm{c_i}) &= \{\mathrm{w_{i,k}}, \forall \mathrm{w_{i,k}} \in \mathrm{L}, \mathrm{i} < \mathrm{k} \leq \mathrm{n}\}, \\
\mathrm{M}(\mathrm{c_i}) &= \{\mathrm{w_{j,k}}, \forall \mathrm{w_{j,k}} \in \mathrm{L}, 1 \leq \mathrm{j} < \mathrm{i} < \mathrm{k} \leq \mathrm{n}\}, \\
\mathrm{E}(\mathrm{c_i}) &= \{\mathrm{w_{j,i}}, \forall \mathrm{w_{j,i}} \in \mathrm{L}, 1 \leq \mathrm{j} < \mathrm{i}\}, \\
\mathrm{S}(\mathrm{c_i}) &= \{\mathrm{c_i}, \exists \mathrm{c_i} \in \mathrm{L}\}.
\end{aligned} \tag{8}$$

Here, L denotes the lexicon we use in this work. Additionally, if a word set is empty, a special word "NONE" is added to the empty word set. An example of this categorization approach is shown in Figure 3. Noted that in this way, not only we can introduce the word embedding, but also no information loss exists since the matching results can be exactly restored from the four word sets of the characters.

Condensing the word sets. After obtaining the "BMES" word sets for each character, each word set is then condensed into a fixed-dimensional vector. In this work, we explored two approaches for implementing this condensation.

The **first** implementation is the intuitive mean-pooling method:

$$\boldsymbol{v}^s(\mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{w \in \mathcal{S}} \boldsymbol{e}^w(w). \tag{9}$$

Here, $\mathcal{S}$ denotes a word set and $\boldsymbol{e}^w$ denotes the word embedding lookup table.

However, as shown in Table 8, the results of empirical studies revealed that this algorithm does not perform well. Therefore, a weighting algorithm is introduced to further leverage the word information. To maintain computational efficiency, we did not opt for a dynamic weighting algorithm like attention. Instead, we propose using the frequency of each word as an indication of its weight. Since the frequency of a word is a static value that can be obtained offline, this can greatly accelerate the calculation of the weight of each word.

Specifically, let $z(w)$ denote the frequency that a lexicon word w occurs in the statistical data, the weighted representation of the word set $\mathcal{S}$ is obtained as follows:

$$\boldsymbol{v}^s(S) = \frac{4}{Z} \sum_{w \in S} z(w) \boldsymbol{e}^w(w), \tag{10}$$

where

$$Z = \sum_{w \in \mathrm{B} \cup \mathrm{M} \cup \mathrm{E} \cup \mathrm{S}} z(w).$$

Here, weight normalization is performed on all words in the four word sets to make an overall comparison.

In this work, the statistical data set is constructed from a combination of training and developing data of the task. Of course, if there is unlabelled data in the task, the unlabeled data set can serve as the statistical data set. In addition, note that the frequency of w does not increase if w is covered by another sub-sequence that matches the lexicon. This prevents the problem in which the frequency of a shorter word is always less than the frequency of the longer word that covers it.

Combining with character representation. The final step is to combine the representations of four word sets into one fix-dimensional feature, and add it to the representation of each character. In order to retain as much information as possible, we choose to concatenate the representations of the four word sets, and the final representation of each character is obtained by:

$$\begin{aligned} \boldsymbol{e}^s(\mathrm{B},\mathrm{M},\mathrm{E},\mathrm{S}) &= [\boldsymbol{v}^s(\mathrm{B});\boldsymbol{v}^s(\mathrm{M});\boldsymbol{v}^s(\mathrm{E});\boldsymbol{v}^s(\mathrm{S})], \\ x^c &\leftarrow [x^c;\boldsymbol{e}^s(\mathrm{B},\mathrm{M},\mathrm{E},\mathrm{S})]. \end{aligned} \tag{11}$$

Here, $\boldsymbol{v}^s$ denotes the weighting function above.

3.3 Sequence Modeling Layer

With the lexicon information incorporated, the character representations are then put into the sequence modeling layer, which models the dependency between characters. Generic architectures for this layer including the bidirectional long-short term memory network(BiLSTM), the Convolutional Neural Network(CNN) and the transformer(Vaswani et al., 2017). In this work, we implemented this layer with a single-layer BiLSTM.

Here, we precisely show the definition of the forward LSTM:

$$\begin{aligned} \begin{bmatrix} i_t \\ f_t \\ o_t \\ \widetilde{c}_t \end{bmatrix} &= \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} \left(W \begin{bmatrix} x_t^c \\ h_{t-1} \end{bmatrix} + b \right), \\ c_t &= \widetilde{c}_t \odot i_t + c_{t-1} \odot f_t, \\ h_t &= o_t \odot \tanh(c_t). \end{aligned} \tag{12}$$

where σ is the element-wise sigmoid function and $\odot$ represents element-wise product. W and b are trainable parameters. The backward LSTM shares the same definition as the forward LSTM yet model the sequence in a reverse order. The concatenated hidden states at the i^{th} step of the forward and backward LSTMs $h_i = [\overrightarrow{h}_i; \overleftarrow{h}_i]$ forms the context-dependent representation of c_i.

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.

3.4 Label Inference Layer

On top of the sequence modeling layer, it is typical to apply a sequential conditional random field (CRF) (Lafferty et al., 2001) layer to perform label inference for the whole character sequence at once:

$$p(y|s;\boldsymbol{\theta}) = \frac{\prod_{t=1}^{n} \phi_t(y_{t-1}, y_t|s)}{\sum_{y' \in \mathcal{Y}_s} \prod_{t=1}^{n} \phi_t(y'_{t-1}, y'_t|s)}. \tag{13}$$

Here, $\mathcal{Y}_s$ denotes all possible label sequences of s, and $\phi_t(y', y|s) = \exp(\boldsymbol{w}_{y',y}^T h_t + b_{y',y})$, where $\boldsymbol{w}_{y',y}$ and $b_{y',y}$ are trainable parameters corresponding to the label pair (y', y), and $\boldsymbol{\theta}$ denotes model parameters. For label inference, it searches for the label sequence y^* with the highest conditional probability given the input sequence s:

$$y^* =_y p(y|s;\boldsymbol{\theta}), \tag{14}$$

which can be efficiently solved using the Viterbi algorithm (Forney, 1973).

4 Experiments

4.1 Experiment Setup

Most experimental settings in this work followed the protocols of Lattice-LSTM (Zhang and Yang, 2018), including tested datasets, compared baselines, evaluation metrics (P, R, F1), and so on. To make this work self-completed, we concisely illustrate some primary settings of this work.

Datasets

The methods were evaluated on four Chinese NER datasets, including OntoNotes (Weischedel et al., 2011), MSRA (Levow, 2006), Weibo NER (Peng

Models	OntoNotes	MSRA	Weibo	Resume
Lattice-LSTM	1×	1×	1×	1×
LR-CNN (Gui et al., 2019)	2.23×	1.57×	2.41×	1.44×
BERT-tagger	2.56×	2.55×	4.45×	3.12×
BERT + LSTM + CRF	2.77×	2.32×	2.84×	2.38×
SoftLexicon (LSTM)	6.15×	5.78×	6.10×	6.13×
SoftLexicon (LSTM) + bichar	6.08×	5.95×	5.91×	6.45×
SoftLexicon (LSTM) + BERT	2.74×	2.33×	2.85×	2.32×

Table 2: Inference speed (average sentences per second, the larger the better) of our method with LSTM layer compared with Lattice-LSTM, LR-CNN and BERT.

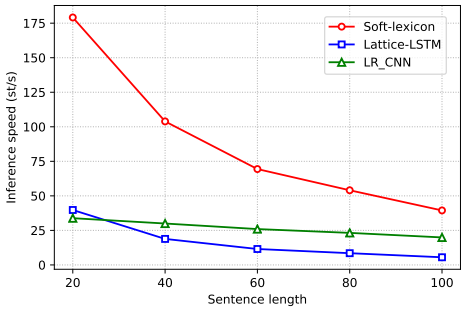

Figure 4: Inference speed against sentence length. We use a same batch size of 1 for a fair speed comparison.

and Dredze, 2015; He and Sun, 2017a), and Resume NER (Zhang and Yang, 2018). OntoNotes and MSRA are from the newswire domain, where gold-standard segmentation is available for training data. For OntoNotes, gold segmentation is also available for development and testing data. Weibo NER and Resume NER are from social media and resume, respectively. There is no gold standard segmentation in these two datasets. Table 1 shows statistic information of these datasets. As for the lexicon, we used the same one as Lattice-LSTM, which contains 5.7k single-character words, 291.5k two-character words, 278.1k three-character words, and 129.1k other words. In addition, the pre-trained character embeddings we used are also the same with Lattice-LSTM, which are pre-trained on Chinese Giga-Word using word2vec.

Implementation Detail

In this work, we implement the sequence-labeling layer with Bi-LSTM. Most implementation details followed those of Lattice-LSTM, including character and word embedding sizes, dropout, embedding initialization, and LSTM layer number. Additionally, the hidden size was set to 200 for small datasets Weibo and Resume, and 300 for larger datasets OntoNotes and MSRA. The initial learning rate was set to 0.005 for Weibo and 0.0015 for the rest three datasets with Adamax (Kingma and Ba, 2014) step rule [2].

4.2 Computational Efficiency Study

Table 2 shows the inference speed of the SoftLexicon method when implementing the sequence modeling layer with a bi-LSTM layer. The speed was evaluated based on the average number of sentences processed by the model per second using a GPU (NVIDIA TITAN X). From the table, we can observe that when decoding with the same batch size (=1), the proposed method is considerably more efficient than Lattice-LSTM and LR-CNN, performing up to 6.15 times faster than Lattice-LSTM. The inference speeds of SoftLexicon(LSTM) with bichar are close to those without bichar, since we only concatenate an additional feature to the character representation. The inference speeds of the BERT-Tagger and SoftLexicon (LSTM) + BERT models are limited due to the deep layers of the BERT structure. However, the speeds of the SoftLexicon (LSTM) + BERT model are still faster than those of Lattice-LSTM and LR-CNN on all datasets.

To further illustrate the efficiency of the SoftLexicon method, we also conducted an experiment to evaluate its inference speed against sentences of different lengths, as shown in Table 4. For a fair comparison, we set the batch size to 1 in all of the compared methods. The results show that the proposed method achieves significant improvement in speed over Lattice-LSTM and LR-CNN when processing short sentences. With the increase of sentence length, the proposed method is consistently faster than Lattice-LSTM and LR-CNN despite the speed degradation due to the recurrent architecture of LSTM. Overall, the proposed SoftLexicon method shows a great advantage over other methods in computational efficiency.

4.3 Effectiveness Study

Tables 3−6[3] show the performances of our method against the compared baselines. In this study, the sequence modeling layer of our method was

[2]Please refer to the attached source code for more implementation detail of this work and access https://github.com/jiesutd/LatticeLSTM for pre-trained word and character embeddings.

[3]In Table 3−5, ∗ indicates that the model uses external labeled data for semi-supervised learning. † means that the model also uses discrete features.

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-based (LSTM)	76.66	63.60	69.52
	+ char + bichar	**78.62**	73.13	75.77
Auto seg	Word-based (LSTM)	72.84	59.72	65.63
	+ char + bichar	73.36	70.12	71.70
No seg	Char-based (LSTM)	68.79	60.35	64.30
	+ bichar + softword	74.36	69.43	71.89
	+ ExSoftword	69.90	66.46	68.13
	+ bichar + ExSoftword	73.80	71.05	72.40
	Lattice-LSTM	76.35	71.56	73.88
	LR-CNN (Gui et al., 2019)	76.40	72.60	74.45
	SoftLexicon (LSTM)	77.28	74.07	75.64
	SoftLexicon (LSTM) + bichar	77.13	**75.22**	**76.16**
	BERT-Tagger	76.01	79.96	77.93
	BERT + LSTM + CRF	81.99	81.65	81.82
	SoftLexicon (LSTM) + BERT	**83.41**	**82.21**	**82.81**

Table 3: Performance on OntoNotes. A model followed by (LSTM) (e.g., Proposed (LSTM)) indicates that its sequence modeling layer is LSTM-based.

implemented with a single layer bidirectional LSTM.

OntoNotes. Table 3 shows results [4] on the OntoNotes dataset, where gold word segmentation is provided for both training and testing data. The methods of the "Gold seg" and the "Auto seg" groups are all word-based, with the former input building on gold word segmentation results and the latter building on automatic word segmentation results by a segmenter trained on OntoNotes training data. The methods used in the "No seg" group are character-based. From the table, we can make several observations. **First**, when gold word segmentation was replaced by automatically generated word segmentation, the F1 score decreases from 75.77% to 71.70%. This reveals the problem of treating the predicted word segmentation result as the true result in the word-based Chinese NER. **Second**, the F1 score of the Char-based (LSTM)+ExSoftword model is greatly improved from that of the Char-based (LSTM) model. This indicates the feasibility of the naive ExSoftword method. However, it still greatly underperforms relative to Lattice-LSTM, which reveals its deficiency in utilizing word information. **Lastly**, the proposed SoftLexicon method outperforms Lattice-LSTM by 1.76% with respect to the F1 score, and obtains a greater improvement of 2.28% combining the bichar feature. It even performs comparably with the word-based methods of the "Gold seg" group, verifying its effectiveness on OntoNotes.

[4] A result in boldface indicates that it is statistically significantly better ($p < 0.01$ in pairwise $t-$test) than the others in the same box.

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
Char-based (LSTM)	90.74	86.96	88.81
+ bichar+softword	92.97	90.80	91.87
+ ExSoftword	90.77	87.23	88.97
+ bichar+ExSoftword	93.21	91.57	92.38
Lattice-LSTM	93.57	92.79	93.18
LR-CNN (Gui et al., 2019)	94.50	92.93	93.71
SoftLexicon (LSTM)	94.63	92.70	93.66
SoftLexicon (LSTM) + bichar	**94.73**	**93.40**	**94.06**
BERT-Tagger	93.40	94.12	93.76
BERT + LSTM + CRF	95.06	94.61	94.83
SoftLexicon (LSTM) + BERT	**95.75**	**95.10**	**95.42**

Table 4: Performance 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**
He and Sun, 2017a	50.60	59.32	54.82
He and Sun, 2017b*	54.50	62.17	58.23
Char-based (LSTM)	46.11	55.29	52.77
+ bichar+softword	50.55	60.11	56.75
+ ExSoftword	44.65	55.19	52.42
+ bichar+ExSoftword	58.93	53.38	56.02
Lattice-LSTM	53.04	62.25	58.79
LR-CNN (Gui et al., 2019)	57.14	**66.67**	59.92
SoftLexicon (LSTM)	**59.08**	62.22	**61.42**
SoftLexicon (LSTM) + bichar	58.12	64.20	59.81
BERT-Tagger	65.77	62.05	63.80
BERT + LSTM + CRF	69.65	64.62	67.33
SoftLexicon (LSTM) + BERT	**70.94**	**67.02**	**70.50**

Table 5: Performance on Weibo. NE, NM and Overall denote F1 scores for named entities, nominal entities (excluding named entities) and both, respectively.

MSRA/Weibo/Resume. Tables 4, 5 and 6 show results on the MSRA, Weibo and Resume datasets, respectively. Compared methods include the best statistical models on these data set, which leveraged rich handcrafted features (Chen et al., 2006; Zhang et al., 2006; Zhou et al., 2013), character embedding features (Lu et al., 2016; Peng and Dredze, 2016), radical features (Dong et al., 2016), cross-domain data, and semi-supervised data (He and Sun, 2017b). From the tables, we can see that the performance of the proposed Softlexion method is significant better than that of Lattice-LSTM and other baseline methods on all three datasets.

Models	P	R	F1
Word-based (LSTM)	93.72	93.44	93.58
+char+bichar	94.07	94.42	94.24
Char-based (LSTM)	93.66	93.31	93.48
+ bichar+softword	94.53	94.29	94.41
+ ExSoftword	95.29	94.42	94.85
+ bichar+ExSoftword	**96.14**	94.72	95.43
Lattice-LSTM	94.81	94.11	94.46
LR-CNN (Gui et al., 2019)	95.37	94.84	95.11
SoftLexicon (LSTM)	95.30	95.77	95.53
SoftLexicon (LSTM) + bichar	95.71	95.77	**95.74**
BERT-Tagger	94.87	**96.50**	95.68
BERT + LSTM + CRF	95.75	95.28	95.51
SoftLexicon (LSTM) + BERT	**96.08**	96.13	**96.11**

Table 6: Performance on Resume.

Models	OntoNotes	MSRA	Weibo	Resume
SoftLexicon (LSTM)	75.64	93.66	61.42	95.53
ExSoftword (CNN)	68.11	90.02	53.93	94.49
SoftLexicon (CNN)	**74.08**	**92.19**	**59.65**	**95.02**
ExSoftword (Transformer)	64.29	86.29	52.86	93.78
SoftLexicon (Transformer)	**71.21**	**90.48**	**61.04**	**94.59**

Table 7: F1 score with different implementations of the sequence modeling layer. ExSoftword is the shorthand of Char-based+bichar+ExSoftword.

4.4 Transferability Study

Table 7 shows the performance of the SoftLexicon method when implementing the sequence modeling layer with different neural architecture. From the table, we can first see that the LSTM-based architecture performed better than the CNN- and transformer- based architectures. In addition, our method with different sequence modeling layers consistently outperformed their corresponding ExSoftword baselines. This confirms the superiority of our method in modeling lexicon information in different neural NER models.

4.5 Combining Pre-trained Model

We also conducted experiments on the four datasets to further verify the effectiveness of SoftLexicon in combination with pre-trained model, the results of which are shown in Tables 3−6. In these experiments, we first use a BERT encoder to obtain the contextual representations of each sequenc, and then concatenated them into the character representations. From the table, we can see that the SoftLexicon method with BERT outperforms the BERT tagger on all four datasets. These results show that the SoftLexicon method can be effectively combined with pre-trained model. Moreover, the results also verify the effectiveness of our method in utilizing lexicon information, which means it can complement the information obtained from the pre-trained model.

Models	OntoNotes	MSRA	Weibo	Resume
SoftLexicon (LSTM)	75.64	93.66	61.42	95.53
- "M" group	75.06	93.09	58.13	94.72
- Distinction	70.29	92.08	54.85	94.30
- Weighted pooling	72.57	92.76	57.72	95.33
- Overall weighting	74.28	93.16	59.55	94.92

Table 8: An ablation study of the proposed model.

4.6 Ablation Study

To investigate the contribution of each component of our method, we conducted ablation experiments on all four datasets, as shown in table 8.

(1) In Lattice-LSTM, each character receives word information only from the words that begin or end with it. Thus, the information of the words that contain the character inside is ignored. However, the SoftLexicon prevents the loss of this information by incorporating the "Middle" group of words. In the " - 'M' group" experiment, we removed the "Middle" group in SoftLexicon, as in Lattice-LSTM. The degradation in performance on all four datasets indicates the importance of the "M" group of words, and confirms the advantage of our method.

(2) Our method proposed to draw a clear distinction between the four "BMES" categories of matched words. To study the relative contribution of this design, we conducted experiments to remove this distinction, i.e., we simply added up all the weighted words regardless of their categories. The decline in performance verifies the significance of a clear distinction for different matched words.

(3) We proposed two strategies for pooling the four word sets in Section 3.2. In the "- Weighted pooling" experiment, the weighted pooling strategy was replaced with mean-pooling, which degrades the performance. Compared with mean-pooling, the weighting strategy not only succeeds in weighing different words by their significance, but also introduces the frequency information of each word in the statistical data, which is verified to be helpful.

(4) Although existing lexicon-based methods like Lattice-LSTM also use word weighting, unlike the proposed Soft-lexion method, they fail to perform weight normalization among all the matched words. For example, Lattice-LSTM only normalizes the weights inside the "B" group or the "E" group. In the "- Overall weighting" experiment, we performed weight normalization inside each

"BMES" group as Lattice-LSTM does, and found the resulting performance to be degraded. This result shows that the ability to perform overall weight normalization among all matched words is also an advantage of our method.

5 Conclusion

In this work, we addressed the computational efficiency of utilizing word lexicons in Chinese NER. To obtain a high-performing Chinese NER system with a fast inference speed, we proposed a novel method to incorporate the lexicon information into the character representations. Experimental studies on four benchmark Chinese NER datasets reveal that our method can achieve a much faster inference speed and better performance than the compared 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 RD Program (No. 2018YFB1005104, 2018YFC0831105, 2017YFB1002104), National Natural Science Foundation of China (No. 61976056, 61532011, 61751201), Shanghai Municipal Science and Technology Major Project (No.2018SHZDZX01), Science and Technology Commission of Shanghai Municipality Grant (No.18DZ1201000, 16JC1420401, 17JC1420200).

References

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

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

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

Jason Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. *Transactions of the Association of Computational Linguistics*, 4(1):357–370.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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

Ruixue Ding, Pengjun Xie, Xiaoyan Zhang, Wei Lu, Linlin Li, and Luo Si. 2019. A neural multi-digraph model for chinese ner with gazetteers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1462–1467.

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 *Natural Language Understanding and Intelligent Applications*, pages 239–250. Springer.

G David Forney. 1973. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278.

Tao Gui, Ruotian Ma, Qi Zhang, Lujun Zhao, Yu-Gang Jiang, and Xuanjing Huang. 2019a. Cnn-based chinese ner with lexicon rethinking. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 4982–4988. AAAI Press.

Tao Gui, Yicheng Zou, Qi Zhang, Minlong Peng, Jinlan Fu, Zhongyu Wei, and Xuan-Jing Huang. 2019b. A lexicon-based graph neural network for chinese ner. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1039–1049.

Hangfeng He and Xu Sun. 2017a. 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: Volume 2, Short Papers*, pages 713–718.

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

Jingzhou He and Houfeng Wang. 2008. Chinese named entity recognition and word segmentation based on character. In *Proceedings of the Sixth SIGHAN Workshop on Chinese Language Processing*.

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

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

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

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

Haibo Li, Masato Hagiwara, Qi Li, and Heng Ji. 2014. Comparison of the impact of word segmentation on name tagging for chinese and japanese. In *LREC*, pages 2532–2536.

Liyuan Liu, Jingbo Shang, Xiang Ren, Frank Xu, Huan Gui, Jian Peng, and Jiawei Han. 2018. Empower sequence labeling with task-aware neural language model. *AAAI Conference on Artificial Intelligence*.

Wei Liu, Tongge Xu, Qinghua Xu, Jiayu Song, and Yueran Zu. 2019. An encoding strategy based word-character lstm for chinese ner. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2379–2389.

Zhangxun Liu, Conghui Zhu, and Tiejun Zhao. 2010. Chinese named entity recognition with a sequence labeling approach: based on characters, or based on words? In *Advanced intelligent computing theories and applications. With aspects of artificial intelligence*, pages 634–640. Springer.

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

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

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

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84.

Dianbo Sui, Yubo Chen, Kang Liu, Jun Zhao, and Shengping Liu. 2019. Leverage lexical knowledge for chinese named entity recognition via collaborative graph network. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3821–3831.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

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

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

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

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

Yue Zhang and Jie Yang. 2018. Chinese ner using lattice lstm. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL), 1554-1564.*

Hai Zhao and Chunyu Kit. 2008. Unsupervised segmentation helps supervised learning of character tagging for word segmentation and named entity recognition. In *Proceedings of the Sixth SIGHAN Workshop on Chinese Language Processing*.

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.