Research Article

Min Lin, Yanyan Xu*, Chenghao Cai, Dengfeng Ke, and Kaile Su

A lattice-transformer-graph deep learning model for Chinese named entity recognition

https://doi.org/10.1515/jisys-2022-2014
received May 08, 2022; accepted January 25, 2023

Abstract: Named entity recognition (NER) is the localization and classification of entities with specific meanings in text data, usually used for applications such as relation extraction, question answering, etc. Chinese is a language with Chinese characters as the basic unit, but a Chinese named entity is normally a word containing several characters, so both the relationships between words and those between characters play an important role in Chinese NER. At present, a large number of studies have demonstrated that reasonable word information can effectively improve deep learning models for Chinese NER. Besides, graph convolution can help deep learning models perform better for sequence labeling. Therefore, in this article, we combine word information and graph convolution and propose our Lattice-Transformer-Graph (LTG) deep learning model for Chinese NER. The proposed model pays more attention to additional word information through position-attention, and therefore can learn relationships between characters by using lattice-transformer. Moreover, the adapted graph convolutional layer enables the model to learn both richer character relationships and word relationships and hence helps to recognize Chinese named entities better. Our experiments show that compared with 12 other state-of-the-art models, LTG achieves the best results on the public datasets of Microsoft Research Asia, Resume, and WeiboNER, with the $F1$ score of 95.89%, 96.81%, and 72.32%, respectively.

Keywords: Chinese named entity recognition, lattice, transformer, graph convolutional networks

MSC 2020: 68T50

1 Introduction

Named entity recognition (NER) is to extract and classify entities with specific meanings, such as person (PER), location (LOC), and organization (ORG). As an upstream process of a number of natural language processing (NLP) tasks, e.g., relationship extraction, event extraction, and automatic question answering [1], it plays an irreplaceable role. Therefore, in-depth research on NER has high practical value [2].

*** Corresponding author: Yanyan Xu,** School of Information Science and Technology, Beijing Forestry University, Beijing 100083, China, e-mail: xuyanyan@bjfu.edu.cn
Min Lin: School of Information Science and Technology, Beijing Forestry University, Beijing 100083, China, e-mail: cs_linmin@bjfu.edu.cn
Chenghao Cai: School of Information Science and Technology, Beijing Forestry University, Beijing 100083, China, e-mail: chenghao.cai@outlook.com
Dengfeng Ke: School of Information Science, Beijing Language and Culture University, Beijing 100083, China, e-mail: dengfeng.ke@blcu.edu.cn
Kaile Su: Institute for Integrated and Intelligent Systems, Griffith University, South East Queensland 4222, Australia, e-mail: k.su@griffith.edu.au

In earlier studies, NER can be solved using sequence annotation models such as hidden Markov model (HMM), maximum-entropy Markov model (MEMM) [3], and conditional random field (CRF) [4]. With the development of deep learning, convolutional neural network (CNN) and long short-term memory (LSTM) are widely used for NER. For example, the LSTM-CRF architecture has been employed to improve the learning ability of neural networks for NER [5,6]. In addition, TENER [7] uses transformer [8] as the encoder with specific modifications for NER. Nowadays, these methods constitute the fundamental structures for NER.

Besides, BERT [9], which is trained on a large dataset containing different languages, can be used as an enhanced feature extractor for NER. Moreover, since each annotation position can be considered as a node on a graph, graph convolutional network (GCN) is utilized to solve the sequence annotation problem by constructing edges between nodes [10]. Other applications of GCN include text semantic role annotations [11,12], text sequence classification [13], etc. Furthermore, GCN can be concatenated after LSTM to improve NER [14]. However, these methods are for word-based languages, such as English. Since Chinese is a character-based language, for Chinese NER, directly adopting these methods cannot achieve good performances.

As a character-based language, in Chinese, there are no clear boundaries between named entities and other characters. For example, the Chinese sentence "北京市长安街" which means "Beijing Chang'an Street," has two location-type named entities "北京市" (Beijing City) and "长安街" (Chang'an Street), but there are no obvious boundaries between these two words. If a model can segment the sentence into "北京市 长安街" it can recognize these two location-type named entities "北京市" and "长安街" more easily. However, "市长" which means "Mayor," is a person-type Chinese named entity. If there lack the relationships between the characters "京" and "市" and "长" and "安" it is possible that this sentence is incorrectly split into "北京 市长 安街" Therefore, both the relationships between words and those between characters are crucial for Chinese NER.

Zhu and Wang try to obtain word information from input sentences directly through a convolutional attention network, using the convolution kernel to help the model segment word boundaries [15] (CAN-NER), but the lengths of Chinese words are variable, so this method has great limitations. Since a Chinese sentence is a character sequence, it is helpful to utilize a lexicon as external knowledge to bring more relationships and linguistic features to the model. Zhang and Yang not only train a lexicon for Chinese NER but also propose a new model to make good use of this lexicon [16] (Lattice-LSTM). However, Lattice-LSTM only adds word information from the lexicon after the last character of a word, so it fails to learn the inner relationships between characters. With regard to other studies, pre-trained language models based on the transformer architecture have made progress in NLP. For example, LEBERT [17] and ZEN [18] adapt BERT to make it more suitable for Chinese NLP, and BERT-wmm [19] obtains a better way to train Chinese pre-trained language models. There are studies which use pre-trained language models as the language embedding extractor [20,21] (DGLSTM-CRF, GAT). However, these Chinese pre-trained language models always pay attention on words rather than characters, so they cannot learn the inner relationships between characters, either.

To learn the inner relationships between characters, the FLAT-Lattice structure [22] is proposed, which not only allows the neural network to extract more linguistic features from the lexicon but also uses relative position coding to enable the transformer model [8] to learn the relationships between different input positions, that is, the inner relationships. Besides, in ref. [23] (LGN), researchers try to use the graph neural network with the lexicon, and in ref. [24] (PLT), researchers try to use the transformer. Moreover, the lattice structure is reconstructed in ref. [22] (FLAT) through self-attention in the transformer model to extract more linguistic and relationship features from the lexicon. Although these studies can learn the inner relationships between characters, they ignore the relationships between words.

In order to learn both the relationships between words and those between characters, in this article, we propose a lattice-transformer-graph (LTG) deep learning model, which uses lattice-transformer to learn inner relationships between characters and modifies GCN to learn not only relationships between characters but also relationships between words from additional word information. Furthermore, as BERT has been adapted to process Chinese NER tasks better by integrating lattice word information [17,18], LTG can also be combined with BERT. We then conduct experiments on four public datasets to demonstrate the effectiveness of LTG. Consequently, our LTG model with using BERT achieves the best results by comparing with 12 other state-of-the-art neural network architectures, including BiLSTM [5], TENER [7], CAN-NER [15], Lattice-LSTM [16], LGN [23], PLT [24], FLAT [22], LEBERT [17], ERINE [25], ZEN [18], DGLSTM-CRF [20], and GAT [21].

2 Proposed methods

2.1 Structure of the LTG model

Our proposed LTG model is illustrated in Figure 1. The position-attention part works on characters and additional words, and these characters and additional words are extracted from the input text to make the LTG model achieve more information. The outputs of Position-Attention are sent to Transformer-Encoder [8] to extract the features of the input text. After that, we use Bi-GCN [11], whose graph is built based on the positions of the additional words, to make the model know more about the structure of the input words. Finally, we put the output features of Bi-GCN into CRF to obtain labels.

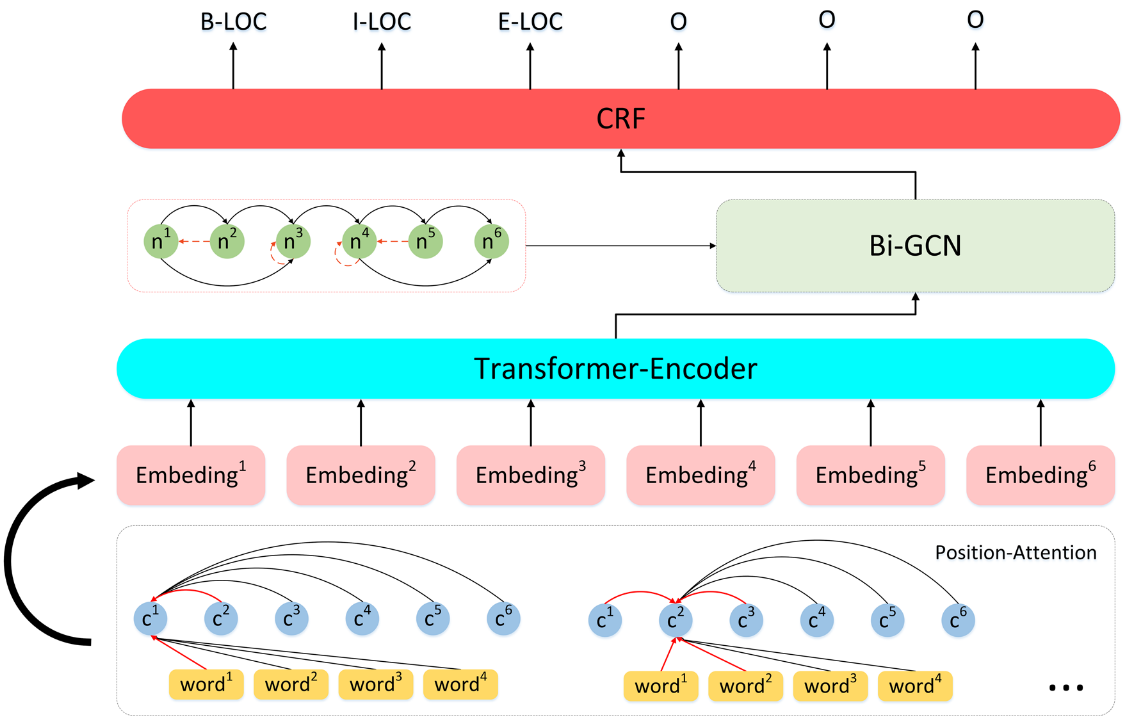
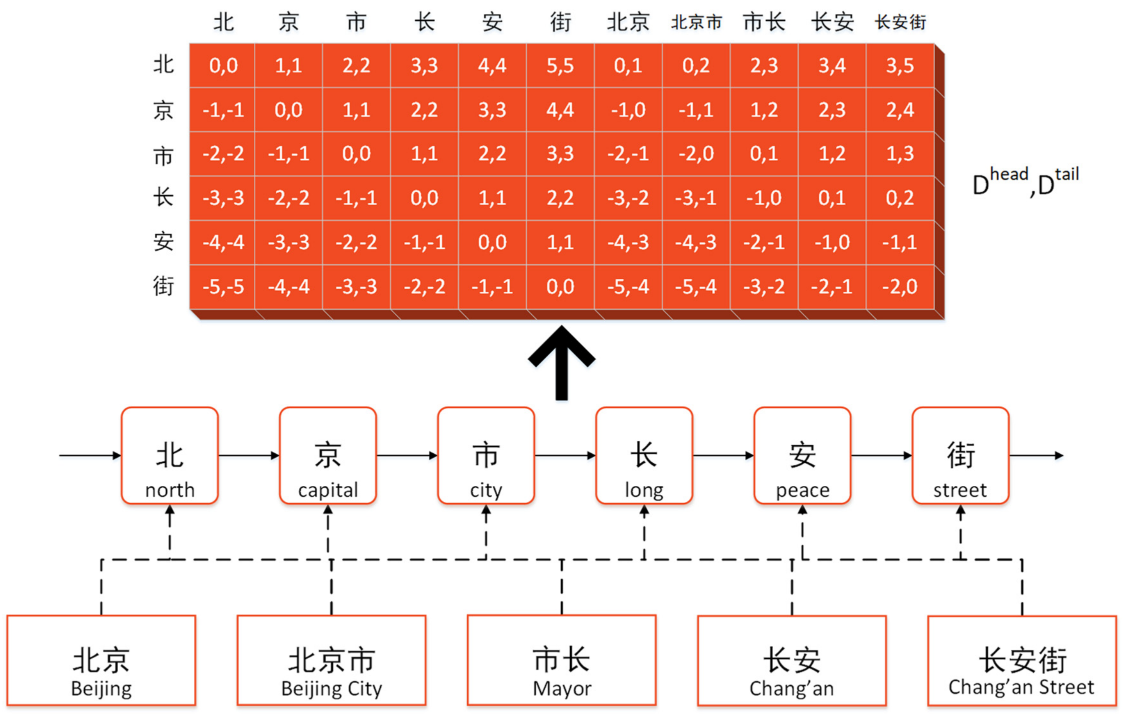


Figure 1: The structure of the LTG model. The part at the bottom (in the black dashed box) is Position-Attention, which helps the model to consider relevant relationships, shown by the red arrows, or irrelevant relationships, shown by the black arrows. The Position-Attention can help the model extract more linguistic features from the input sequence and external words, and also the relationship features between them. The output embeddings of Position-Attention are sent to Transformer-Encoder to obtain features. Transformer-Encoder can handle long-distance dependencies better than RNN. After that, Bi-GCN calculates these features using the graph (the part in the red dashed box) built based on the additional words, and the graph can help the model learn deeper linguistic features and relationship features. Finally, the model sends the outputs of Bi-GCN to CRF to obtain labels.

In the LTG model, we use transformer-encoder [8] and Bi-GCN [11] to extract features. The Bi-GCN can make the neural network to focus more on the relationships between additional words and characters. The graph construction method used in our study is based on the positions of the additional words, which can learn the context relationships between words and characters better. The reason of using additional word information is that, in the process of Chinese NER, Chinese characters are nodes, but most of the named entities are words which consist of several Chinese characters rather than a single Chinese character. By making the model pay more attention to the positions of additional words, the performance of Chinese NER can be improved.

2.2 Lattice

For the reason that lattice significantly improves Chinese NER, our study uses Flat-Lattice [22] as additional information. The input sequence of the model is composed of Chinese characters. In addition, there are words contained in the lexicon in all subsequences of the sequence. These characters and words are regarded as the lattice input, and transforming them into embeddings is completed by the trained lexicon. As shown in Figure 2, the input sentence is "北京市长安街 (Beijing Chang'an Street)", which contains five words in the lexicon: "北京 (Beijing)," "北京市 (Beijing city)," "市长 (Mayor)," "长安 (Chang'an)," and "长安街 (Chang'an Street)," so there are five additional words in the input. In order to conduct parallel computing during training this model, the relationships between words and characters will be considered before they are computed by the next layer. This study uses $W^{d}_{b,e}$ to represent these words, where b and e are



	北	京	市	长	安	街	北京	北京市	市长	长安	长安街
北	0,0	1,1	2,2	3,3	4,4	5,5	0,1	0,2	2,3	3,4	3,5
京	-1,-1	0,0	1,1	2,2	3,3	4,4	-1,0	-1,1	1,2	2,3	2,4
市	-2,-2	-1,-1	0,0	1,1	2,2	3,3	-2,-1	-2,0	0,1	1,2	1,3
长	-3,-3	-2,-2	-1,-1	0,0	1,1	2,2	-3,-2	-3,-1	-1,0	0,1	0,2
安	-4,-4	-3,-3	-2,-2	-1,-1	0,0	1,1	-4,-3	-4,-3	-2,-1	-1,0	-1,1
街	-5,-5	-4,-4	-3,-3	-2,-2	-1,-1	0,0	-5,-4	-5,-4	-3,-2	-2,-1	-2,0

Figure 2: The input text is "北京市长安街 (Beijing Chang'an Street)," whose subsequences "北京 (Beijing)," "北京市 (Beijing City)," "市长 (Mayor)," "长安 (Chang'an)," and "长安街 (Chang'an Street)" are in the lexicon. The subsequences are the additional words of the input. We use Position-Attention to consider every word and every character. The number pair (D_{head}, D_{tail}) above represents the distance not only between two characters but also between a character and a word. All of them will be sent to the Position-Attention layer.

the beginning position and the end position of the word, and d is the dimension of the word embeddings. For example, "北京 (Beijing)" can be shown as $W_{1,2}^{d}$, and "长安街 (Chang'an Street)" can be shown as $W_{4,6}^{d}$.

2.3 Position-attention

Since the model considers the relationships between each word and each character, irrelevant words and characters will also be calculated together, such as "北 (north)" and "长安 (Chang'an)," which will undoubtedly influence the subsequent procedure. Therefore, Position-Attention is needed to distinguish the relevant and irrelevant relationships. In order to obtain the Position-Attention of the context of words and characters, we calculate the position embedding representing the position relationships between them. First, we calculate the relative positions of words and characters through (1) and (2):

$$D_{ij}^{\mathrm{head}} = \mathrm{HEAD}_{\mathrm{word}}[i] - \mathrm{POS}_{\mathrm{char}}[j], \tag{1}$$

$$D_{ij}^{\mathrm{tail}} = \mathrm{TAIL}_{\mathrm{word}}[i] - \mathrm{POS}_{\mathrm{char}}[j], \tag{2}$$

where $\mathrm{HEAD}_{\mathrm{word}}$ indicates the position of the first character in a word, and $\mathrm{TAIL}_{\mathrm{word}}$ indicates the position of the last one, and $\mathrm{POS}_{\mathrm{char}}$ indicates the position of a single character. Combining (1) and (2), we know that when $D_{ij}^{\mathrm{head}} < 0$ and $D_{ij}^{\mathrm{tail}} < 0$, a character is the pre-context of a word; while $D_{ij}^{\mathrm{head}} \leq 0$ and $D_{ij}^{\mathrm{tail}} \geq 0$ means a character is in a word, and $D_{ij}^{\mathrm{head}} < 0$ and $D_{ij}^{\mathrm{tail}} < 0$ means a character is the post-context of a word. To facilitate parallel computing, characters can be passed through $\mathrm{HEAD}_{\mathrm{word}}[i] = \mathrm{TAIL}_{\mathrm{word}}[i]$, and the relationship between characters is $D_{\mathrm{char}}^{\mathrm{head}} = D^{c2c} = D_{\mathrm{char}}^{\mathrm{tail}}$. (1) and (2) can then be simplified to (3):

$$D_{ij}^{c2c} = \mathrm{POS}_{\mathrm{char}}[i] - \mathrm{POS}_{\mathrm{char}}[j]. \tag{3}$$

The relative position embeddings of characters can be calculated by (3). Our study follows [8] to obtain the position embeddings of distances, shown by (4) and (5):

$$p_{D}^{(2k)} = \sin\left(\frac{D}{10{,}000^{2k/d_{\mathrm{model}}}}\right), \tag{4}$$

$$p_{D}^{(2k+1)} = \cos\left(\frac{D}{10{,}000^{2k/d_{\mathrm{model}}}}\right), \tag{5}$$

where D is the relative position calculated by (1), (2), and (3), and k is the current dimension, and d_{model} is the dimension of the model. Each dimension of (4) and (5) corresponds to the sinusoidal curve [8]. The results of the position embeddings are the embeddings of the relative position distances. The relative position embedding P between word/character i and j is calculated by:

$$P_{ij} = \mathrm{ReLU}\big(W_r\big(p_{D^{\mathrm{head}}} \oplus p_{D^{\mathrm{tail}}}\big)\big). \tag{6}$$

Through the position embedding of the relationship between each two words/characters, the output feature containing context and word information can be obtained. The feature will be sent to the transformer layer, and the model obtains the input containing context information between words and characters. The calculation of transformer is defined as follows:

$$\mathrm{Attention}(Q, K, V) = \left(\frac{QK^{T}}{\sqrt{d_{\mathrm{head}}}}\right)V, \tag{7}$$

$$[Q, K, V] = (E_{\mathrm{pre}} + E_{\mathrm{post}})[W_q, W_k, W_v], \tag{8}$$

where E_{pre} represents the original input embedding or the output of the previous transformer-encoder layer, and E_{post} is the embedding calculated by $E_{\mathrm{pre}}P$. $W_q, W_k, W_v \in \mathbb{R}^{d_{\mathrm{model}} \cdot d_{\mathrm{head}}}$ are learnable parameters, and $d_{\mathrm{model}} = H \cdot d_{\mathrm{head}}$, where H is the number of heads of multihead-attention, and d_{head} is the dimension of each head.

2.4 Bidirectional graph CNN

2.4.1 Building graphs with word information

We use GCN to help the model learn the relationships between adjacent nodes, so we construct the graphs with the positions of words and the inner relationships of characters in words. A graph is represented as $G = (V, E)$, where V is the node set, that is, the character set, and E is the edge set.

Algorithm 1 Construct graph G^1

```
    Input: The start positions and lengths of all additional words from the input text
    Output: $G^1$, the graph that helps the model learn the relationships between characters
1:  for each $word \in Words$ do
2:    $i^1 \Leftarrow Start_{word}$
3:    for $j^1 = i^1 + 1$ to $i^1 + Length_{word}$ do
4:      construct an edge $e^1_{(i^1,j^1)}$ from $i^1$ to $j^1$
5:    end for
6:  end for
```

We construct two graphs, G^1 representing the inner relationships in words and G^2 representing the relationships between words. In G^1, if two nodes satisfy $start \leq i^1 \leq j^1 \leq end$, we construct an edge $e^1_{i^1j^1} = (v^1_i, v^1_j)$. The construction of G^1 is detailed in Algorithm 1. Benefiting from G^1, Bi-GCN helps the network learn the context of characters in words and extract more linguistic information from words.

Algorithm 2 Construct graph G^2

```
     Input: The additional words from the input text
     Output: $G^2$, the graph that helps the model learn the relationships between words
1:   $Word_{now} \Leftarrow NULL$
2:   for $Pos = 0$ to $Length_{sentence}$ do
3:     Generate the set of the words as $Word_{pos}$ whose first character is at Pos
4:     if $Word_{pos}$ is not empty then
5:       if the number of words in $Word_{pos}$ is greater than 1 then
6:         Push the words in $Word_{pos}$ into Stack in descending order of the lengths of these words
7:         while $Stack$ is not empty do
8:           if $Word_{now} = =NULL$ then
9:           $Word_{now} \Leftarrow Stack_{top}$
10:          else
11:          construct an edge $e^2_{(W_n,S_t)}$ from $Word_{now}$ to $Stack_{top}$
12:          $Word_{now} \Leftarrow Stack_{top}$
13:          end if
14:          $Stack_{top}$ out
15:        end while
16:      else
```

```
17:  if $Word_{now}$ = =$NULL$ then
18:          $Word_{now} \Leftarrow Word_{pos}$
19:        else
20:          construct an edge $e^2_{(W_n,W_p)}$ from $Word_{now}$ to $Word_{pos}$
21:          $Word_{now} \Leftarrow Word_{pos}$
22:        end if
23:      end if
24:    end if
25:  end for
```

G^2 is the graph of the relationships between words, which is constructed by Algorithm 2. The edge of G^2 is represented as $e^2_{i^2j^2} = (v^2_i, v^2_j)$, where $0 \le i < j < \mathrm{Num_{words}}$. Bi-GCN uses G^2 to make the model extract more context information between words. For example, "北京(Beijing)" and "北京市(Beijing City)" have similar linguistic features, and both of them have strong correlations with "长安街(Chang'an Street)." However, some words are useless, such as "市长(Major)." Context information can also help the model reduce these useless words' weights, that is, the redundant information can be removed. G^1 and G^2 of this example "北京市长安街 (Beijing Chang'an Street)" are shown in Figure 3.

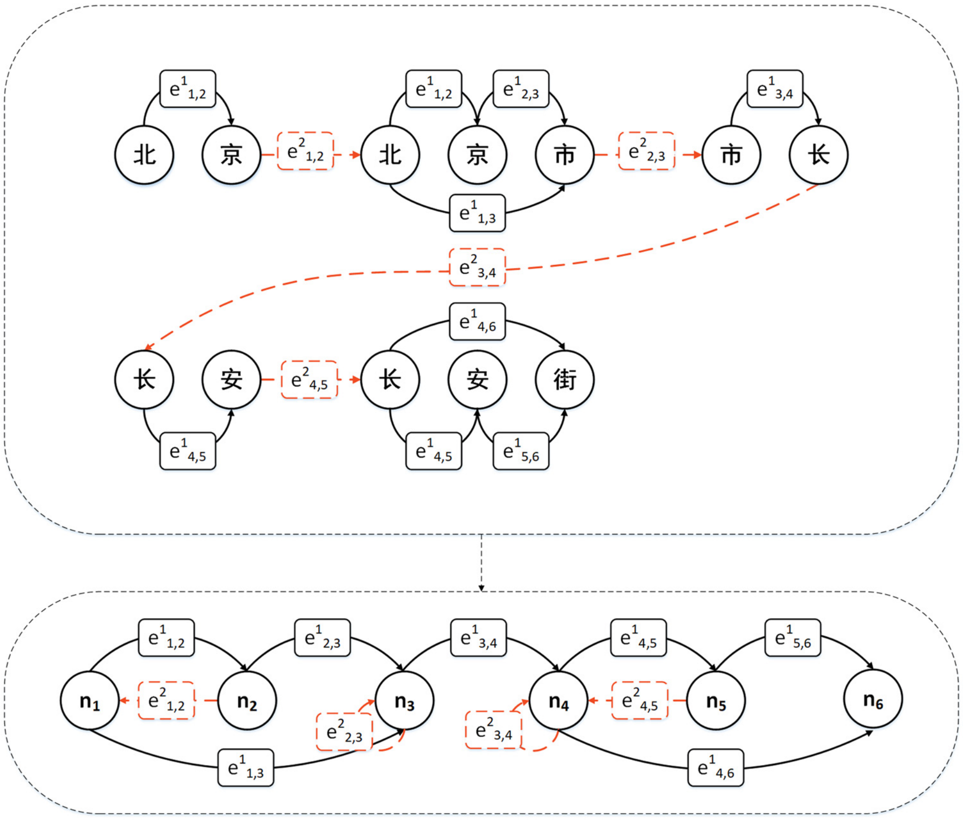


Figure 3: As the upper figure shows, among words like "北京 (Beijing)" and , "北京市 (Beijing City)," there are edges from one internal node to the next internal node, which is represented by $e^1_{b,e}$ in G^1, shown with the black solid lines. $e^2_{b,e}$ in G^2 shown with the red dotted line represents an edge from a word to its next word. In order to make the model learn the context information at the same time, we use Bi-GCN, which only needs to build the forward edges in the graph. The upper figure can be simplified to the below figure, and nodes n_1, n_2, n_3, n_4, n_5, and n_6 represent the characters "北", "京", "市", "长", "安", "街" respectively.

2.4.2 Bidirectional graph convolutional layer

Algorithms 1 and 2 only show the construction of edges from front nodes to rear nodes. Actually, we use a Bi-GCN [11] layer which changes the directed graph into an undirected graph and make the model to learn context information. Given a directed graph $G = (V, E)$, and the input with length L: $X = x_1, x_2, \ldots, x_L$, the feature f_i for each node i can be obtained by the following equations:

$$\overrightarrow{f_i} = \mathrm{ReLU}\left(\sum_{e_{ij}\in E}(\overrightarrow{W_f}x_j + \overrightarrow{b_f})\right), \tag{9}$$

$$\overleftarrow{f_i} = \mathrm{ReLU}\left(\sum_{e_{ji}\in E}(\overleftarrow{W_f}x_j + \overleftarrow{b_f})\right), \tag{10}$$

$$f_i = [\overrightarrow{f_i}; \overleftarrow{f_i}], \tag{11}$$

$W_f \in \mathbb{R}^{d_x \cdot d_f}$ includes learnable parameters, and $b_f \in \mathbb{R}^{d_f}$ is the offset. d_x is the dimension of the input, and d_f is the dimension of the hidden layers in GCN. ReLU is a nonlinear activation function. Concatenating the output features from the two graphs to represent the output features of the GCN layer is shown as follows:

$$f = W_c(f^1 \oplus f^2) + b_c, \tag{12}$$

$W_c \in \mathbb{R}^{2d_f \cdot d_f}$ includes learnable parameters, and $b_c \in \mathbb{R}^{d_f}$ represents the offset. f^1 and f^2 are the graph features of G^1 and G^2, respectively. After using Bi-GCN to obtain the output features $F = f_1, f_2, \ldots, f_L$ of every node, CRF [4] can compute labels through these features.

3 Experimental setup

3.1 Datasets

We use four public datasets to verify the model, including Ontonotes 5.0 [26], Microsoft Research Asia (MSRA) [27], Resume [16], and WeiboNER [28,29], which are illustrated in Table 1.

Table 1: Sizes of the datasets

Dataset	Train	Test	Dev
Ontonotes 5.0	9.5M	1.4M	1.2M
MSRA	13.5M	1.5M	1.2M
Resume	1.1M	118K	132K
WeiboNER	524K	104K	107K

Ontonotes 5.0: The dataset includes texts in English, Arabic, and Chinese, covering six areas: radio conversations, radio news, magazine articles, news-wires, telephone conversations, and web blogs [26]. In this study, only Chinese text sequences and NER labels in this dataset are selected.

MSRA: It is a simplified Chinese dataset provided by MSRA for word segmentation and NER [27].

Resume: The dataset is constructed from resume data of Sina Finance. The data established by Sina Finance include the resumes of executives of listed companies in the Chinese stock market. The team randomly selected 1,027 resume abstracts and manually annotated 8 types of named entities using the YEDDA system [30]. The labeling consistency between them is 97.1%, and the complete dataset is called Resume [16].

WeiboNER: The construction of the dataset follows the DEFT ERE guidelines and includes four main named entity types: person names, organization names, place names, and geopolitics names. The corpus includes 1,890 messages collected from Weibo species between November 2013 and December 2014. The WeiboNER dataset is constructed by manually labeling these messages [29].

3.2 Evaluation metrics

Let TP, FP, and FN be the numbers of true positive, false positive, and false negative examples, respectively. Then the precision $P = \text{TP}/(\text{TP} + \text{FP})$ and the recall rate $R = \text{TP}/(\text{TP} + \text{FN})$ can be used to evaluate the performance of an NER model.

Although P and R are equally important in judging the performance, they are contradictory indicators. By predicting all the uncertain labels as non-named entities, P will increase. However, in this case, R decreases. On the contrary, by predicting all the uncertain labels as named entities, R will increase, but in this case, P decreases. Therefore, in our experiments, we use their combination $F1 = 2 * P * R/(P + R)$ as the evaluation metric, which can balance P and R.

3.3 State-of-the-art models for comparison

In our experiments, we assess the performance of our model LTG against those of 12 other state-of-the-art models.
– BiLSTM [5]: it simply uses bi-directional LSTM and CRF.
– TENER [7]: it modifies the transformer for NER tasks.
– CAN-NER [15]: it uses CNN to segment word boundaries.
– Lattice-LSTM [16]: it uses the lattice structure to add word information from the lexicon.
– LGN [23]: it uses the graph neural network with the lexicon.
– PLT [24]: it uses the transformer with the lexicon.
– FLAT [22]: it reconstructs the lattice structure to utilize the lexicon better.
– LEBERT [17]: it uses the lexicon to enhance BERT.
– ERINE [25]: it uses informative entities to enhance language presentation.
– ZEN [18]: it uses n-gram representations to enhance the Chinese text encoder.
– DGLSTM-CRF [20]: it uses BERT as the feature extractor with a dependency-guide LSTM-CRF structure.
– GAT [21]: it uses BERT as the feature extractor with a structure enhancing entity boundary detection.

Among these models, BiLSTM, TENER, and CAN-NER are without a lexicon, and are used as the baseline models, so that we can demonstrate the advantage of using lattice in Chinese NER. Also, LTG is compared with a number of recent well-performed models without using BERT, including Lattice-LSTM, LGN, PLT, and FLA. In addition, our model using BERT as the feature extractor, that is, BERT+LTG, is compared with the models also using BERT or with a large number of parameters, such as LEBERT, ERINE, ZEN, DGLSTM-CRF, and GAT.

4 Experimental results

4.1 Comparison with state-of-the-art models

In order to show the effectiveness of LTG and BERT+LTG, we compare them with all the 12 state-of-the-art models mentioned in 3.3. We conduct experiments on the public datasets of Ontonotes 5.0, MSRA, Resume, and WeiboNER, mentioned in 3.1.

Table 2 details the experimental results of these models. It demonstrates that our model BERT+LTG achieves the best results on multiple datasets, especially on WeiboNER. Since WeiboNER is collected from Weibo, a social networking platform, most of the sentences are daily Chinese, and the distribution of named entities is significantly sparse compared with other datasets. Therefore, GCN plays an obvious role in facilitating the network understand word information and the relationship between word information and context. At the same time, Position-Attention also helps the network pay more attention to key information. Hence, LTG performs better on WeiboNER than on other datasets.

Table 2: Comparison of LTG with other advanced models

Model	**Ontonotes 5.0** (%)	**MSRA** (%)	**Resume** (%)	**WeiboNER** (%)
BiLSTM [5], 2018	—	91.87	94.41	56.75
TENER [7], 2019	—	93.01	95.25	58.39
CAN-NER [15], 2019	—	92.97	94.94	59.31
Lattice-LSTM [16], 2018	—	93.18	94.46	58.79
LGN [23], 2019	—	93.47	95.37	63.09
PLT [24], 2019	—	93.26	95.40	59.92
FLAT [22], 2020	—	94.35	94.93	63.42
LEBERT [17], 2021	—	95.70	96.08	70.75
ERINE [25], 2019	—	95.32	95.46	68.32
ZEN [18], 2020	—	95.20	95.40	66.71
DGLSTM-CRF [20], 2019	77.40	—	—	—
GAT [21], 2021	**79.53**	—	—	69.50
LTG, ours	74.70	95.06	95.16	66.67
BERT + LTG, ours	78.78	**95.89**	**96.81**	**72.32**

The bold values highlight the best results.

Resume is a dataset from resume data of Sina Finance. For Resume, our LTG model can extract more features from word information to perform better than other models.

However, LTG do not obtain the best F1 score on Ontonotes 5.0. The reason is that LTG relies on the help of word information, but there are too many long distances in this dataset which consequently produce large amount of word information, making it harder for LTG to handle it. Nevertheless, LTG achieves comparable results on this dataset.

MSRA is also a dataset with long distances, but there are only three types of labels in this dataset, so LTG can recognize these labels without extracting a large number of linguistic and relation features from word information. Therefore, it performs best on this dataset.

Furthermore, we aim to build an architecture that can work well both with a pre-trained language model, such as BERT, and without a pre-trained language model. It can be seen from Table 3 that when the application scenario is closer to those of MSRA and Resume, the improvements of using BERT, that is, a large-parameter language model, are not obvious. Since BERT needs a lot of computations, in these scenarios, LTG without BERT is more suitable. On the other hand, when the application scenario is closer to those of Ontonotes 5.0 and WeiboNER, the improvements of using BERT are obvious, and in these scenarios, BERT + LTG is a better choice.

Table 3: Comparison of LTG and BERT + LTG

Model	**Ontonotes 5.0** (%)	**MSRA** (%)	**Resume** (%)	**WeiboNER** (%)
LTG	74.70	95.06	95.16	66.67
BERT+LTG	78.78	95.89	96.81	72.32
Relative improvement	**5.46**	0.87	1.7	**8.47**

The bold values highlight the best results.

4.2 Ablation experiments

In order to demonstrate the functionality of different components in our LTG model, we perform ablation experiments on the dataset of WeiboNER. Table 4 shows the results of ablation experiments.

Table 4: Ablation experiments on WeiboNER

Ablation	Settings	*F*1 (%)	*P* (%)	*R* (%)
Number of Bi-GCN layers	#layers=1	72.32	69.34	**75.58**
	#layers=2	**72.66**	70.63	74.81
	#layers=3	72.56	**72.38**	72.75
Bidirectional and unidirectional GCN	Bidirectional	**72.32**	**69.34**	**75.58**
	Unnidirectional	69.1	63.64	**75.58**
	Without GCN	68.12	67.86	68.38
Ways of embedding	Position-attention	**72.32**	**69.34**	**75.58**
	Lattice only	69.41	65.53	73.78

The bold values highlight the best results.

4.2.1 Number of Bi-GCN layers

In many cases, adding layers will make a deep learning model work better. In this way, we try to add Bi-GCN layers in our model. However, as shown in Table 4, as the number of Bi-GCN layers increases, the performance does not increase. We conclude that the number of Bi-GCN layers makes no obvious difference in our model.

4.2.2 Bidirectional and unidirectional GCN

For the reason that we only build a forward graph for GCN and context information takes an important part in a sequence labeling task, bidirectional-GCN is necessary for the model to learn context information. From Table 4, we see that bidirectional GCN can bring significant improvements, compared with unidirectional GCN and not using GCN. GCN uses the graph to extract more relationship and linguistic features, and the bidirectional structure helps to reduce the loss of context features.

4.2.3 Ways of embedding

Position-Attention helps the model learn more about the context information between words and characters. For comparison, we use another way to process the embedding called “lattice only,” which means adding lattice as Lattice-LSTM [16]. The experiment results in Table 4 show that Position-Attention proposed in this article performs much better than “lattice only.” At the same time, these experimental results also prove that obtaining more relationship features between the word information and the input sequence can facilitate the model to perform better.

4.3 Residual experiments

In order to further prove the effectiveness of GCN and the possibility of building a deeper network structure, we conduct residual experiments of GCN on WeiboNER, by adding the output before entering the GCN layer with the output of the GCN layer and send them to the CRF layer for decoding.

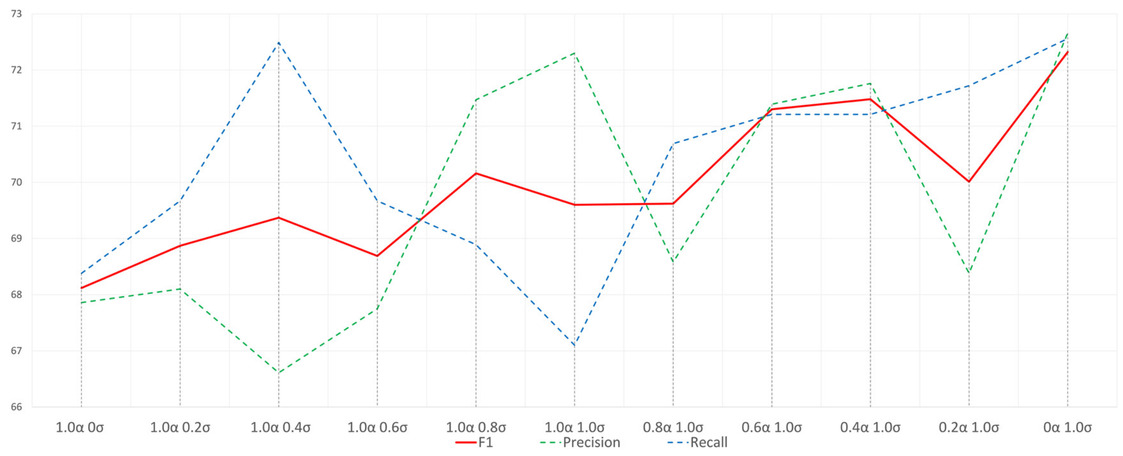


Figure 4: Residual experiments on WeiboNER.

In the experiments, the encoding E_{pre} before passing through the GCN layer and the encoding E_{GCN} after passing through the GCN layer are added under different weights. As illustrated in equation (13), the encoding entering the next layer is as follows:

$$E_{\text{post}} = \alpha E_{\text{pre}} + \sigma E_{\text{GCN}}, \tag{13}$$

where α and σ are the weights of E_{pre} and E_{GCN}, respectively.

Experimental results obtained by adjusting different weights are shown in Figure 4. It is shown that when $\alpha = 0$ and $\sigma = 1.0$, the network performs best for *F1*, *Precision* and *Recall*. That is, the network without residual calculation performs best. At the same time, all the experimental results with the GCN layer are better than those without the GCN layer, demonstrating that the coding without GCN contains redundant information, which interferes with the coding results through the GCN layer after addition. Therefore, it further shows that the GCN layer can help the network extract key information.

As for the residual structure, although the bidirectional GCN retains the contextual features, the structure of the output features is not the same as the sequential structure of the original features, so adding the residuals of these two parts will bring confusing information. Therefore, the LTG model cannot build a deeper GCN through the residual structure, and hence cannot extract richer features for Chinese NER.

5 Conclusions and future work

In this article, we proposed the LTG model and applied it to Chinese NER. LTG uses Position-Attention to improve the transformer model's ability to learn the relationships between characters and modifies GCN to improve the network's understanding of word information. The experimental results show that LTG achieves the best results so far for Chinese NER on the public datasets of MSRA, Resume, and WeiboNER. Moreover, LTG can be used both with and without BERT.

However, LTG does not work well on Ontonotes 5.0, due to too much redundant information, so in the future, we will try to reduce the redundant information in our model to improve its generalization ability. Besides, since the aim of NER is to serve downstream tasks and the results of the downstream tasks can give feedback to NER [31], we will combine NER more with its downstream tasks in the future work.

Acknowledgment: The authors thank the editor and reviewers for their valuable comments and suggestions.

Funding information: This work was supported by the Fundamental Research Funds for the Central Universities (grant number 2021ZY87).

Conflict of interest: The authors declare that there is no conflict of interests.

References

[1] Chen Y, Xu L, Liu K, Zeng D, Zhao J. Event extraction via dynamic multi-pooling convolutional neural networks. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers); 2015. p. 167–76.
[2] Jin Y, Xie J, Guo W, Luo C, Wu D, Wang R. LSTM-CRF neural network with gated self attention for Chinese NER. IEEE Access. 2019;7:136694–703.
[3] McCallum A, Freitag D, Pereira FC. Maximum entropy Markov models for information extraction and segmentation. In: Icml. United States: Morgan Kaufmann, vol. 17. 2000. p. 591–8.
[4] Lafferty JD, McCallum A, Pereira FC. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: Proceedings of the Eighteenth International Conference on Machine Learning; 2001. p. 282–9.
[5] Panchendrarajan R, Amaresan A. Bidirectional LSTM-CRF for named entity recognition. In: Proceedings of the 32nd Pacific Asia Conference on Language, Information and Computation, Hong Kong, China, December 1–3, 2018; 2018. p. 531–40.
[6] Lample G, Ballesteros M, Subramanian S, Kawakami K, Dyer C. Neural architectures for named entity recognition. In: Knight K, Lopez A, Mitchell M, editors. Human Language Technologies. 2016 Conference of the North American Chapter of the Association for Computational Linguistics; 2016 June 12-17; San Diego (CA, USA).[Sl]: Association for Computational Linguistics (ACL); 2016. p. 260–70. ACL (Association for Computational Linguistics); 2016.
[7] Yan H, Deng B, Li X, Qiu X. TENER: adapting transformer encoder for named entity recognition. 2019. arXiv: http://arXiv.org/abs/arXiv:191104474.
[8] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. In: Guyon I, vonLuxburg U, Bengio S, Wallach HM, Fergus R, Vishwanathan SVN, et al., editors. Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017. Long Beach, CA, USA; 2017. p. 5998–6008. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.
[9] Kenton JDMWC, Toutanova LK. BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of NAACL-HLT; 2019. p. 4171–86.
[10] Kipf TN, Welling M. Semi-supervised classification with graph convolutional networks. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net; 2017. https://openreview.net/forum?id=SJU4ayYgl.
[11] Marcheggiani D, Titov I. Encoding sentences with graph convolutional networks for semantic role labeling. In: Palmer M, Hwa R, Riedel S, editors. Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9–11, 2017. Association for Computational Linguistics; 2017. p. 1506–15. doi: https://doi.org/10.18653/v1/d17-1159.
[12] Yao L, Mao C, Luo Y. Graph convolutional networks for text classification. In: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019. AAAI Press; 2019. p. 7370–7. doi: https://doi.org/10.1609/aaai.v33i01.33017370.
[13] Peng H, Li J, He Y, Liu Y, Bao M, Wang L, et al. Large-scale hierarchical text classification with recursively regularized deep graph-CNN. In: Champin P, Gandon F, Lalmas M, Ipeirotis PG, editors. Proceedings of the 2018 Conference on , WWW 2018, Lyon, France, April 23–27, 2018. ACM; 2018. p. 1063–72. doi: 10.1145/3178876.3186005.
[14] Luo Y, Zhao H. Bipartite flat-graph network for nested named entity recognition. In: Jurafsky D, Chai J, Schluter N, Tetreault JR, editors. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020. Association for Computational Linguistics; 2020. p. 6408–18. doi: 10.18653/v1/2020.acl-main.571.
[15] Zhu Y, Wang G. CAN-NER: convolutional attention network for Chinese named entity recognition. In: Burstein J, Doran C, Solorio T, editors. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2–7, 2019, Volume 1 (Long and Short Papers). Association for Computational Linguistics; 2019. p. 3384–93. doi: 10.18653/v1/n19-1342.
[16] Zhang Y, Yang J. Chinese NER using Lattice LSTM. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers); 2018. p. 1554–64.
[17] Liu W, Fu X, Zhang Y, Xiao W. Lexicon enhanced Chinese sequence labeling using BERT adapter. In: Zong C, Xia F, Li W, Navigli R, editors. Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th

International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1–6, 2021. Association for Computational Linguistics; 2021. p. 5847–58. doi: 10.18653/v1/2021.acl-long.454.

[18] Diao S, Bai J, Song Y, Zhang T, Wang Y. ZEN: pre-training Chinese text encoder enhanced by N-gram representations. In: Cohn T, He Y, Liu Y, editors. Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020. vol. EMNLP 2020 of Findings of ACL. Association for Computational Linguistics; 2020. p. 4729–40. doi: 10.18653/v1/2020.findings-emnlp.425.

[19] Cui Y, Che W, Liu T, Qin B, Wang S, Hu G. Revisiting pre-trained models for Chinese natural language processing. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings. Online: Association for Computational Linguistics; 2020. p. 657–68. https://www.aclweb.org/anthology/2020.findings-emnlp.58.

[20] Jie Z, Lu W. Dependency-guided LSTM-CRF for named entity recognition. In: Inui K, Jiang J, Ng V, Wan X, editors. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3–7, 2019. Association for Computational Linguistics; 2019. p. 3860–70. doi: 10.18653/v1/D19-1399.

[21] Chen C, Kong F. Enhancing entity boundary detection for better Chinese named entity recognition. In: Zong C, Xia F, Li W, Navigli R, editors. Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 2: Short Papers), Virtual Event, August 1-6, 2021. Association for Computational Linguistics; 2021. p. 20–25. doi: 10.18653/v1/2021.acl-short.4.

[22] Li X, Yan H, Qiu X, Huang X. FLAT: Chinese NER using Flat-Lattice transformer. In: Jurafsky D, Chai J, Schluter N, Tetreault JR, editors. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5–10, 2020. Association for Computational Linguistics; 2020. p. 6836–42. doi: 10.18653/v1/2020.acl-main.611.

[23] Gui T, Zou Y, Zhang Q, Peng M, Fu J, Wei Z, et al. A Lexicon-based graph neural network for Chinese NER. In: Inui K, Jiang J, Ng V, Wan X, editors. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019. Association for Computational Linguistics; 2019. p. 1040–50. doi: 10.18653/v1/D19-1096.

[24] Mengge X, Bowen Y, Tingwen L, Yue Z, Erli M, Bin W. Porous lattice-based transformer encoder for Chinese ner. 2019. arXiv: http://arXiv.org/abs/arXiv:191102733.

[25] Zhang Z, Han X, Liu Z, Jiang X, Sun M, Liu Q. ERNIE: enhanced language representation with informative entities. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. Florence, Italy: Association for Computational Linguistics; 2019. p. 1441–51. https://aclanthology.org/P19-1139.

[26] Pradhan S, Moschitti A, Xue N, Ng HT, Björkelund A, Uryupina O, et al. Towards robust linguistic analysis using OntoNotes. In: Proceedings of the Seventeenth Conference on Computational Natural Language Learning. Bulgaria: Association for Computational Linguistics; 2013. p. 143–52. https://aclanthology.org/W13-3516.

[27] Levow G. The third international Chinese language processing Bakeoff: word segmentation and named entity recognition. In: Ng HT, Kwong OOY, editors. Proceedings of the Fifth Workshop on Chinese Language Processing, SIGHAN@COLING/ACL 2006, Sydney, Australia, July 22–23, 2006. Association for Computational Linguistics; 2006. p. 108–17. https://aclanthology.org/W06-0115/.

[28] Peng N, Dredze M. Named entity recognition for Chinese social media with jointly trained embeddings. In: Màrquez L, Callison-Burch C, Su J, Pighin D, Marton Y, editors. Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015. The Association for Computational Linguistics; 2015. p. 548–54. doi: 10.18653/v1/d15-1064.

[29] He H, Sun X. F-score driven max margin neural network for named entity recognition in Chinese social media. In: Lapata M, Blunsom P, Koller A, editors. Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers. Association for Computational Linguistics; 2017. p. 713–8. doi: 10.18653/v1/e17-2113.

[30] Yang J, Zhang Y, Li L, Li X. YEDDA: A lightweight collaborative text span annotation tool. In: Proceedings of ACL 2018, System Demonstrations. Melbourne, Australia: Association for Computational Linguistics; 2018. p. 31–6. https://aclanthology.org/P18-4006.

[31] Yan Z, Zhang C, Fu J, Zhang Q, Wei Z. A partition filter network for joint entity and relation extraction. In: Moens M, Huang X, Specia L, Yih SW, editors. Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7–11 November, 2021. Association for Computational Linguistics; 2021. p. 185–97. doi: 10.18653/v1/2021.emnlp-main.17.