A Chinese Named Entity Recognition Method Based on Textual Information Perception Fusion

Wanting Ji, Lei Zhang, and Baoyan Song(✉)

Liaoning University, No. 66, Chongshan Middle Road, Huanggu District, Shenyang, Liaoning, China
bysong@lnu.edu.cn

Abstract. Named entity recognition (NER) aims to identify the entities with specific meanings from text, which is an important basic tool in many fields such as information extraction, question answering, and machine translation. In real-world, to perform named entity recognition tasks conveniently and quickly, it requires NER methods can quickly and accurately identify entities from the input text after being trained by a small training set for subsequent operations. The existing Chinese NER methods are usually based on BERT + LSTM + CRF models. However, when the training set is small, the recognition accuracy of these methods is relatively low. To solve this problem, this paper proposes a named entity recognition method based on Textual Information Perception Fusion (TIPF). It fully extracts the global features of the Chinese text through the Textual Information Memory Perception module, and fully fuses the global features and local features of the Chinese text using the Textual Information Adaptive Fusion module, so that the Chinese NER can be realized quickly and accurately. The proposed method is tested on several public datasets. Experimental results show that, compared with the existing methods, TIPF can achieve a higher recognition accuracy after training with a smaller training set.

Keywords: named entity recognition · deep neural network · natural language processing · data integration

1 Introduction

Named entity recognition (NER) is one of the fundamental tasks in natural language processing, which aims to identify entities with specific meanings from text [1], such as person, location, organization, etc. In the process of multi-source data integration, NER is often used in integration operations such as data cleaning and similar join to determine whether different records in one or more data sources are described as the same entity [2].

Recent research on Chinese NER mainly focuses on deep neural network-based methods [3], especially the methods based on BERT + LSTM + CRF [4] model, which contains a Bidirectional Encoder Representation from Transformers (BERT) [5] to encode the input text, a Long Short-Term Memory (LSTM) network [6] to extract features from the encoded text, and a Conditional Random Field (CRF) [7] to label entities

X. Yang et al. (Eds.): ADMA 2023, LNAI 14179, pp. 196–210, 2023.
https://doi.org/10.1007/978-3-031-46674-8_14

based on the extracted features. For example, Liu Bingyang [8] proposed a Chinese NER method incorporating global word boundary features, which suffered from the problem of word boundary errors when using the double-character combination boundary features for location recognition. Wei Zhu [9] proposed a Lex-BERT model that introduced entity type information. Although the model introduces entity information into the underlying part of BERT, which can improve recognition accuracy, the model relies heavily on high-quality vocabulary with entity type information during the recognition process, and the portability is poor.

More recently, the Lattice LSTM network [10] is proposed. It creates a thesaurus through text segmentation, and then encodes the character sequence of the text to be recognized and matches the latent words in the text according to the created lexicon. The above process is based on characters and makes full use of the word and word sequence information in the text. However, its accuracy will decrease rapidly when the number of training samples is reduced.

In the real world, some applications utilize NER methods to meet users' needs. In these applications [11], to interact with users in (near) real-time, the number of training samples of NER methods is usually reduced to reduce the training time, to ensure the high interactivity of the model. However, for the existing NER methods, reducing the training time by reducing the number of training samples will reduce the recognition accuracy of the model and even lead to under-fitting. Therefore, how to reduce the number of training samples under the premise of ensuring recognition accuracy is a problem to be solved.

To solve the above problems, this paper proposes a method based on Textual Information Perception Fusion (TIPF) for Chinese named entity recognition. It consists of two modules (a Textual Information Memory Perception module and a Textual Information Adaptive Fusion module) to extract the text-level information (global features) and the word-level information (local features). The proposed method is tested on several public datasets such as People's Daily 2014, Weibo NER, Boson, etc. Experimental results show that TIPF outperforms the state-of-the-art methods for Chinese NER, and can achieve a higher recognition accuracy than other methods after short-term training with a small number of training samples.

The main contributions of this paper are summarized as follows.

(1) A Textual Information Perception Fusion based method is proposed for Chinese named entity recognition.
(2) Based on a key-value memory network, a Textual Information Memory Perception module is proposed to extract text features to improve recognition accuracy.
(3) Based on an attention mechanism, a Textual Information Adaptive Fusion module is proposed to calculate the proportion of each information branch and improve the accuracy of information fusion.

This paper contains five chapters. Section 1 introduces the background knowledge related to named entity recognition and discusses the existing Chinese named entity recognition methods. Section 2 introduces the proposed method. Section 3 introduces the experimental environment used, and discusses and analyses the experimental results. Section 4 concludes the whole paper and gives future research directions.

2 The Proposed Method

This paper proposes a Chinese NER method based on Textual Information Memory Perception named TIPF. As shown in Fig. 1, TIPF consists of four components. The left part is the Textual Information Memory Perception module, the right part is the Lattice LSTM module, and the middle part is the Textual Information Adaptive Fusion module, the top part is the conditional random field module. Specifically, after encoding the input text sequence in character units, it is sent to the Lattice LSTM module for feature extraction, and each character generates a hidden vector that incorporates matching word information. Then, the hidden vector corresponding to each character is sent to the Textual Information Memory Perception module, stored in the key-value memory network, and the hidden vectors of all characters that are the same as the character are extracted from the key-value memory network to generate the corresponding context information. The context information of the character and the hidden vector of the character are calculated by the Textual Information Adaptive Fusion module to calculate their respective weights, and the two are fused by weighted summation. Finally, the hidden vector sequence fused with context information is sent to the conditional random field module for entity labeling. In this way, TIPF can integrate the feature information of the same words in different contexts, make full use of text-level information, and realize the memory perception of text information from a global perspective, to improve the accuracy of NER.

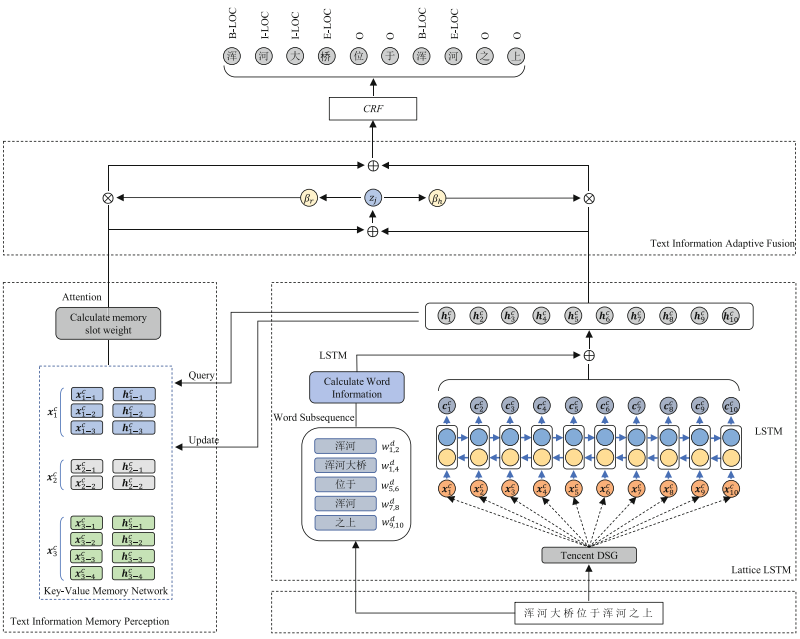

Fig. 1. TIPF method structure

2.1 Word Representation and Word Embedding

In this paper, we use an open-source Chinese word embedding data of Tencent AI Lab to encode the input text sequence [12]. To improve the labeling accuracy, character and word embedding will be fine-tuned during the training of NER.

For the text to be recognized by the named entity, let its length be m, and denote it as a sequence of characters: $C = \{c_1, c_2, \ldots, c_m\}$. For the j th character c_j in this sequence, its embedding is denoted as $\boldsymbol{x}_j^c$:

$$\boldsymbol{x}_j^c = e^c(c_j) \tag{1}$$

where e^* denotes an embedding lookup table. Using a Chinese word segmenter to divide the large raw text into words to generate the corresponding lexicon D, and form a sub-sequence of all words in the lexicon D that match the characters in the character sequence, which is represented by the form of $w_{b,j}^d$. As shown in Fig. 2, the starting values of b and j are 1, which respectively represent the start and the end subscript of the word in the character sequence. The word embedding of $w_{b,j}^d$ is:

$$\boldsymbol{x}_{b,j}^w = e^w\left(w_{b,j}^d\right) \tag{2}$$

Both character sequences and word sub-sequences are used as the input of the model, and the BIOES tagging scheme [13] is used for NER tagging.

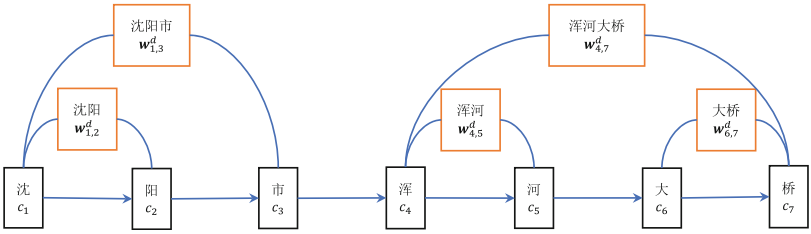

Fig. 2. Word representation

2.2 Lattice LSTM Module

This paper is based on the character-based LSTM model, and the LSTM network used to calculate the hidden vector of characters can be expressed as:

$$\begin{bmatrix} \boldsymbol{i}_j^c \\ \boldsymbol{o}_j^c \\ \boldsymbol{f}_j^c \\ \tilde{\boldsymbol{c}}_j^c \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} \left(\boldsymbol{W}^{c^T} \begin{bmatrix} \boldsymbol{x}_j^c \\ \boldsymbol{h}_{j-1}^c \end{bmatrix} + \boldsymbol{b}^c \right) \tag{3}$$

$$\boldsymbol{c}_j^c = \boldsymbol{f}_j^c \odot \boldsymbol{c}_{j-1}^c + \boldsymbol{i}_j^c \odot \tilde{\boldsymbol{c}}_j^c \tag{4}$$

$$\boldsymbol{h}_j^c = \boldsymbol{o}_j^c \odot \tan \mathrm{h}\left(\boldsymbol{c}_j^c\right) \tag{5}$$

where $\mathbf{c}_j^c$ serves to record recurrent information flow from the beginning of the sentence to c_j. When there is no related matching word in c_j, $\mathbf{c}_j^c$ can be calculated by the formula (4). $\mathbf{h}_j^c$ denotes the hidden vector corresponding to each character c_j. $\boldsymbol{i}^*$, $\boldsymbol{o}^*$ and $\mathbf{f}^*$ denote a set of input, output and forget gates, σ represents the *sigmoid* function, $\boldsymbol{W}^*$ and $\boldsymbol{b}^*$ represent the weight and bias of the model.

When there are related matching words in c_j, the cell vector (word information) of the related matching words is calculated by the simplified LSTM model. Since the tagging process is based on characters, there is no need to calculate the hidden vector corresponding to the matching word when calculating the word information. Therefore, the output gate is removed based on the general LSTM model, and only $\boldsymbol{c}_{b,j}^w$ is used to represent the recurrent state of $\boldsymbol{x}_{b,j}^w$ from the beginning of the sentence, finally, a simplified LSTM model is obtained. The word embedding $\boldsymbol{x}_{b,j}^w$ of the matching word is sent to the model to calculate the word information $\boldsymbol{c}_{b,j}^w$.

$$\begin{bmatrix} \boldsymbol{i}_{b,j}^w \\ \boldsymbol{f}_{b,j}^w \\ \tilde{\boldsymbol{c}}_{b,j}^w \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \tan h \end{bmatrix} \left(\boldsymbol{W}^{w\mathrm{T}} \begin{bmatrix} \boldsymbol{x}_{b,j}^w \\ \boldsymbol{h}_b^c \end{bmatrix} + \boldsymbol{b}^w \right) \tag{6}$$

$$\boldsymbol{c}_{b,j}^w = \boldsymbol{f}_{b,j}^w \odot \boldsymbol{c}_b^c + \boldsymbol{i}_{b,j}^w \odot \tilde{\boldsymbol{c}}_{b,j}^w \tag{7}$$

By combining character input and word input, each $\mathbf{c}_j^c$ has more sources of information flow. In Fig. 3, the input sources of $\boldsymbol{c}_7^c$ include $\boldsymbol{x}_7^c$, $\boldsymbol{c}_{6,7}^w$, and $\boldsymbol{c}_{4,7}^w$.

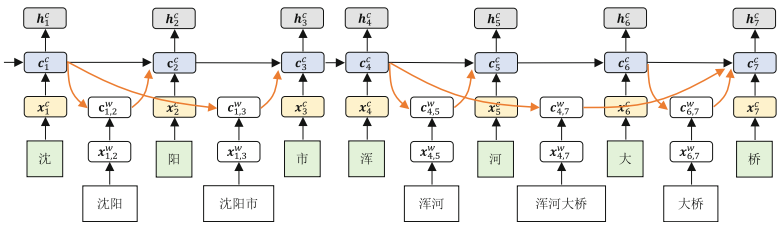

Fig. 3. Lattice LSTM module

Each $\boldsymbol{c}_{b,j}^w$ is linked with $b \in \left\{ b' | w_{b',j}^d \in \mathbb{D} \right\}$ to the cell $\mathbf{c}_j^c$. Since the importance of each word information $\boldsymbol{c}_{b,j}^w$ to $\mathbf{c}_j^c$ is different, the gate control unit $\boldsymbol{i}_{b,j}^c$ is used to control the proportion of each $\boldsymbol{c}_{b,j}^w$, and the gate control unit $\boldsymbol{i}_j^c$ is used to control the contribution of the temporary unit state $\tilde{\boldsymbol{c}}_j^c$ to $\mathbf{c}_j^c$ corresponding to $\boldsymbol{x}_j^c$. Finally, normalize the $\boldsymbol{i}_{b,j}^c$ and $\boldsymbol{i}_j^c$ to $\boldsymbol{\alpha}_{b,j}^c$ and $\boldsymbol{\alpha}_j^c$. The calculation method is as follows:

$$\boldsymbol{i}_{b,j}^c = \sigma \left(\boldsymbol{W}^{l\mathrm{T}} \begin{bmatrix} \boldsymbol{x}_j^c \\ \boldsymbol{c}_{b,j}^w \end{bmatrix} + \boldsymbol{b}^l \right) \tag{8}$$

$$\boldsymbol{\alpha}_{b,j}^{c} = \frac{exp\left(\boldsymbol{i}_{b,j}^{c}\right)}{exp\left(\boldsymbol{i}_{j}^{c}\right) + \sum_{b' \in \left\{b'' | w_{b'',j}^{d} \in \mathbb{D}\right\}} exp\left(\boldsymbol{i}_{b',j}^{c}\right)} \tag{9}$$

$$\boldsymbol{\alpha}_{j}^{c} = \frac{exp\left(\boldsymbol{i}_{j}^{c}\right)}{exp\left(\boldsymbol{i}_{j}^{c}\right) + \sum_{b' \in \left\{b'' | w_{b'',j}^{d} \in \mathbb{D}\right\}} exp\left(\boldsymbol{i}_{b',j}^{c}\right)} \tag{10}$$

Therefore, when there are related matching words in c_j, $\mathbf{c}_j^c$ can be calculated as:

$$\boldsymbol{c}_{j}^{c} = \sum_{b \in \left\{b' | w_{b',j}^{d} \in \mathbb{D}\right\}} \boldsymbol{\alpha}_{b,j}^{c} \odot \boldsymbol{c}_{b,j}^{w} + \boldsymbol{\alpha}_{j}^{c} \odot \tilde{\boldsymbol{c}}_{j}^{c} \tag{11}$$

Finally, the LSTM hidden vector $\boldsymbol{h}_j^c$ corresponding to c_j is calculated as:

$$\boldsymbol{h}_{j}^{c} = \boldsymbol{o}_{j}^{c} \odot \tan h\left(\boldsymbol{c}_{j}^{c}\right) \tag{12}$$

Applying the input sequence $\boldsymbol{X} = \left(\boldsymbol{x}_1^c, \boldsymbol{x}_2^c, \ldots, \boldsymbol{x}_m^c\right)$ to the bidirectional Lattice LSTM model, respectively, the bidirectional hidden vectors $\overrightarrow{\boldsymbol{h}_1^c}, \overrightarrow{\boldsymbol{h}_2^c}, \ldots, \overrightarrow{\boldsymbol{h}_m^c}$ and $\overleftarrow{\boldsymbol{h}_1^c}, \overleftarrow{\boldsymbol{h}_2^c}, \ldots, \overleftarrow{\boldsymbol{h}_m^c}$ can be obtained. For the character c_j, the hidden vector $\boldsymbol{h}_j^c$ after the fusion of related matching word information can be expressed as: $\boldsymbol{h}_j^c = \left[\overrightarrow{\boldsymbol{h}_j^c}, \overleftarrow{\boldsymbol{h}_j^c}\right]$.

2.3 Text Information Memory Perception Module

In this paper, a key-value memory network [14] is constructed for sequence feature extraction to obtain text-level information. The key-value memory network is based on the key memory cache for addressing, and uses the value memory cache for reading, which has a good effect on storing global information.

First, the LSTM hidden vector $\boldsymbol{h}_j^c$ corresponding to each c_j is used as an extra knowledge source to help prediction [15], and the corresponding key-value memory network is generated. Key-value memory networks store text-level information based on "key-value" memory components. The character embedding $\boldsymbol{x}_j^c$ of the character c_j is used as the key, and the corresponding LSTM hidden vector $\boldsymbol{h}_j^c$ is used as the value. During entity recognition, it is stored in memory slots.

Memory slots are defined as pairs of vectors $((\boldsymbol{k}_1, \boldsymbol{v}_1) \ldots (\boldsymbol{k}_m, \boldsymbol{v}_m))$, where m denotes the number of memory slots. So the same character occupies in many different memory slots because of changing representations and embeddings under different contexts. In each training epoch, each memory slot in the key-value memory network will be updated once. For example, if the value of the hidden vector corresponding to the i th character changes after calculation in a new training epoch, the value of the i th memory slot in the corresponding key-value memory network will be rewritten to the calculated value.

2.4 Attention Mechanism

For the i th character P_i in the entire training set, this paper all the contextualized representations for this character P_i in the key-value memory network through an inverted index [16] that find a subset $M_{P_i} : \left(\left(\boldsymbol{k}_{s_1}, \boldsymbol{v}_{s_1}\right) \ldots \left(\boldsymbol{k}_{s_T}, \boldsymbol{v}_{s_T}\right)\right)$ of size T, and the inverted index records the positions of the same character in the key-value memory network. T represents the number of occurrences of character P_i among the training set. In Table 1, the 16th character "河" can be found through the inverted index to find that its subscript set in the key-value memory network is [4, 10, 16,…].

Table 1. Training instance

1. 沈(0)阳(1)市(2)浑(3)河(4)大(5)桥(6)。
2. 位(7)于(8)浑(9)河(10)之(11)上(12)。
3. 沟(13)通(14)浑(15)河(16)两(17)岸(18)交(19)通(20)。
4. …
Invert index: 河 [4,10,16…]

For the character P_i, the contribution of the memory slots in the contextualized representation M_{P_i} to the character P_i in the labelling process will be different depending on the context. Therefore, this paper needs to use the attention mechanism [17] to calculate the weight of the contribution of each memory slot in M_{P_i} to P_i.

For each character, in this paper, the key $\boldsymbol{k}_j \in \left[\boldsymbol{k}_{s_1}; \ldots; \boldsymbol{k}_{s_J}\right]$ of the memory slot in the corresponding subset of the character is used as the attention key, the value $\boldsymbol{v}_j \in \left[\boldsymbol{v}_{s_1}; \ldots; \boldsymbol{v}_{s_J}\right]$ of the memory slot is used as the attention value, and the character embedding $\boldsymbol{x}_{q_i}$ of the queried character serves as the attention query $\boldsymbol{q}_i$. This paper uses Formula (13) to calculate the correlation degree $\boldsymbol{u}_{ij} = o\left(\boldsymbol{q}_i, \boldsymbol{k}_j\right)$ of $\boldsymbol{q}_i$ and $\boldsymbol{k}_j$:

$$o\left(\boldsymbol{q}_i, \boldsymbol{k}_j\right) = \boldsymbol{q}_i \boldsymbol{k}_j^T \tag{13}$$

Finally, according to u_{ij}, the contribution weight $\boldsymbol{\alpha}_{ij}$ of the memory slot whose key is $\boldsymbol{k}_j$ to the character embedding $\boldsymbol{x}_{q_i}$ can be calculated:

$$\boldsymbol{\alpha}_{ij} = \frac{exp\left(\boldsymbol{u}_{ij}\right)}{\sum_{z=1}^{T} exp\left(\boldsymbol{u}_{iz}\right)} \tag{14}$$

For the character c_i, the text-level information $\boldsymbol{r}_i$ related to it can be calculated by weighted summation:

$$\boldsymbol{r}_i = \sum_{j=1}^{J} \boldsymbol{\alpha}_{ij} \boldsymbol{v}_j \tag{15}$$

2.5 Text Information Adaptive Fusion

In order to integrate the text-level information $\boldsymbol{r}_i$ of the current character c_i with its corresponding hidden vector $\boldsymbol{h}_i^c$ more perfectly, the integration ratio of the two should be determined according to the context of each character. In this way, the more important parts of the two are better extracted, and the less important representations are discarded. In this paper, a Textual Information Adaptive Fusion module is designed, and its goal is to enable the neural network to adaptively adjust the proportion of the two fusions according to the recognition results. The basic idea is to use gates to control the information flow from text-level information $\boldsymbol{r}_i$ and hidden vector $\boldsymbol{h}_i^c$, so that these branches carry information of different scales into the final hidden state. To achieve this goal, the gates need to integrate information from all branches, resulting in better fusion results [18].

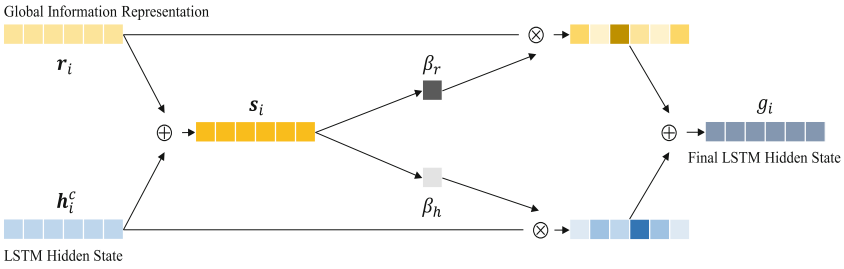

Fig. 4. Textual Information Adaptive Fusion module

The Fig. 4 shows the structure of the Textual Information Adaptive Fusion module. First, the results of the text-level information $\boldsymbol{r}_i$ and the hidden vector $\boldsymbol{h}_i^c$ are preliminary fused through element-wise summation to generate the feature descriptor $\boldsymbol{s}_i$ for precise and adaptive selection guidance:

$$\boldsymbol{s}_i = \boldsymbol{h}_i^c + \boldsymbol{r}_i \tag{16}$$

Soft attention is used to adaptively select different dimensions of information, and β_r and β_h denote soft attention of $\boldsymbol{r}_i$ and $\boldsymbol{h}_i^c$ respectively.

$$\beta_r = \frac{exp\left(\boldsymbol{W}_r \boldsymbol{s}_i^T\right)}{exp\left(\boldsymbol{W}_r \boldsymbol{s}_i^T\right) + exp\left(\boldsymbol{W}_h \boldsymbol{s}_i^T\right)} \tag{17}$$

$$\beta_h = \frac{exp\left(\boldsymbol{W}_h \boldsymbol{s}_i^T\right)}{exp\left(\boldsymbol{W}_r \boldsymbol{s}_i^T\right) + exp\left(\boldsymbol{W}_h \boldsymbol{s}_i^T\right)} \tag{18}$$

The final hidden state $\boldsymbol{g}_i$ is obtained by multiplying $\boldsymbol{r}_i$, $\boldsymbol{h}_i^c$ by their respective soft attention and adding them:

$$\boldsymbol{g}_i = \beta_r \cdot \boldsymbol{r}_i + \beta_h \cdot \boldsymbol{h}_i^c \tag{19}$$

2.6 Conditional Random Field Module

The sequence $\boldsymbol{g}_1, \boldsymbol{g}_2, \ldots, \boldsymbol{g}_m$ is sent to the standard CRF layer for decoding, and the probability of obtaining a label sequence $y = l_1, l_2, \ldots, l_m$ is:

$$P(y|s) = \frac{exp\left(\sum_i \left(\boldsymbol{W}_{\text{CRF}}^{l_i} \boldsymbol{g}_i + \boldsymbol{b}_{\text{CRF}}^{(l_{i-1}, l_i)}\right)\right)}{\sum_{y'} exp\left(\sum_i \left(\boldsymbol{W}_{\text{CRF}}^{l_{i'}} \boldsymbol{g}_i + \boldsymbol{b}_{\text{CRF}}^{(l_{i-1'}, l_{i'})}\right)\right)} \tag{20}$$

where y' represents an arbitrary label sequence. This paper uses the first-order Viterbi algorithm [19] to calculate the score of each label sequence, and the set of label sequences with the highest score will be used as the sequence labelling result. The loss function can be expressed as:

$$L = \sum_{i=1}^{N} log(P(y_i|s_i)) + \frac{\lambda}{2}\Theta^2 \tag{21}$$

where $\{(s_i, y_i)\}|_{i=1}^{N}$ denotes a set of manually labelled training data, λ is the L_2 regularization parameter, and Θ represents the parameter set.

3 Experiments

To demonstrate the accuracy and effectiveness of the proposed method, experiments are conducted on the People's Daily 2014 dataset, Weibo NER dataset and Boson dataset. The comparison methods include the current best-performing Lattice LSTM [10] method, the Lex-BERT [9] method, and the classic BiLSTM-CRF method [20]. In the same experimental environment, compare the performance of the TIPF method and existing methods and analyze the experimental results. The precision, recall and F1 value of are used as evaluation criteria to evaluate the performance of the method.

The specific information of the software and hardware environment of the training process is, CPU: i5-10210U; memory: 16GB; operating system: macOS 11; Python version: 3.8.12; PyTorch version: 1.10.0. The hyper-parameter settings of TIPF are shown in Table 2.

Table 2. Hyper-parameter values

Parameters	Value	Parameters	Value
char emb size	200	LSTM hidden	200
char dropout	0.1	LSTM layer	1
learning rate *lr*	0.015	*lr* decay	0.05

Table 3. Statistics of datasets

Datasets	Type	Train	Dev	Test
People's Daily 2014 (2K)	sentence	2.0k	0.3k	0.3k
	char	186.4k	20.6k	16.9k
People's Daily 2014 (22W)	sentence	22.0w	3.3w	3.3w
	char	1822.9w	268.8w	272.4w
Weibo NER	sentence	1.3k	0.3k	0.3k
	char	73.7k	14.5k	14.8k
Boson	sentence	1.8k	0.3k	0.3k
	char	412.3k	67.4k	65.2k

3.1 Experimental Datasets

This paper conducts experiments on the People's Daily 2014, Weibo NER and Boson datasets. The statistical results for each dataset are shown in Table 3. Due to the large corpus of the People's Daily, in order to simulate the environment of a small data set, this paper randomly intercepts part of the data for experiments (People's Daily 2014(2K)), and conducts experiments on the full data set (People's Daily 2014(22W)) of the People's Daily to verify the effectiveness of the Textual Information Memory Perception module in the TIPF method.

3.2 Experimental Results on Several Datasets

Table 4 shows the precision, recall and F1 value of TIPF and comparative methods for Chinese NER on the People's Daily 2014 (2K) dataset. Compared with other methods, TIPF achieves the highest performance on each item.

Table 5 shows the experimental results on the People's Daily 2014 dataset. It can be seen that after increasing the amount of training data, the recognition accuracy of the Lattice LSTM model has improved, but it is lower than the performance of the TIPF method trained with 2k pieces of data. This shows that in the case of a small dataset, TIPF can use the Textual Information Memory Perception module to fuse global and local features, thereby achieving better recognition accuracy.

Table 6 shows the recognition accuracy, recall and F1 value of TIPF and contrasting methods on the Weibo NER dataset. Compared with other methods, TIPF achieves the highest or equal performance on each item.

Table 7 shows the experimental results on the Boson dataset. Compared with other methods, TIPF achieved the highest F1 value, recognition accuracy and recall.

To sum up, comparing the experimental results of Weibo NER, People's Daily 2014 and Boson in Table 4, 5, 6 and 7, it is found that the performance improvement effect of the TIPF method on the People's Daily 2014 data set is more obvious. By analyzing the data samples in these three datasets, it is found that the People's Daily 2014 data set is based on the official news corpus of the People's Daily, and some of the sentences

Table 4. Results on the People's Daily 2014(2K) dataset

Models	Precision	Recall	F1
Lattice LSTM [10]	87.61	75.49	81.10
BiLSTM-CRF [20]	76.63	79.05	77.82
Lex-BERT [9]	89.77	76.28	82.48
TIPF	**91.97**	**80.53**	**85.87**

Table 5. Results on the People's Daily 2014 dataset

Models	Data Type	Precision	Recall	F1
Lattice LSTM [10]	People's Daily 2014 (2K)	87.61	75.49	81.10
Lattice LSTM [10]	People's Daily 2014 (22W)	90.86	78.25	84.08
TIPF	People's Daily 2014 (2K)	**91.97**	**80.53**	**85.87**

Table 6. Results on the Weibo NER dataset

Models	Precision	Recall	F1
Lattice LSTM [10]	51.69	34.52	41.39
BiLSTM-CRF [20]	49.32	35.16	41.05
Lex-BERT [9]	52.63	35.48	42.39
TIPF	**56.53**	**35.94**	**43.94**

Table 7. Results on the Boson dataset

Models	Precision	Recall	F1
Lattice LSTM [10]	72.58	65.32	68.76
BiLSTM-CRF [20]	70.00	64.24	67.00
Lex-BERT [9]	72.19	65.91	68.91
TIPF	**75.12**	**68.25**	**71.52**

to be recognized in the training set are all from the same news corpus, and the context of the sentences to be recognized is more closely related. In the Weibo NER dataset and the Boson dataset, there are fewer sentences from the same corpus, which makes the contextual connection of the sentences to be recognized relatively weak, resulting in different overall performances of TIPF on these three datasets.

The different performances of the TIPF method on the three datasets also illustrate the effectiveness of this method. In the People's Daily 2014 data set, since many sentences to be recognized are from the same news corpus and have close contextual relationships, the Textual Information Memory Perception module in the TIPF method can accurately store and memorize this context information, thereby improving the recognition efficiency of the named entity recognition. On the Weibo NER dataset and the Boson dataset, although the entity recognition accuracy of the TIPF method is generally lower than that on the People's Daily 2014 dataset, it is still higher than that of the existing methods. This shows that the TIPF method can fuse the global and local features of the data as much as possible to achieve better entity recognition results even in the face of data with weak contextual connections.

3.3 Time Analysis

The above experimental results show that the Textual Information Memory Perception module can achieve better recognition accuracy on small data sets. But for this paper, the ultimate purpose of reducing the number of training sets is to reduce the training time of the model. For this reason, we record the relevant time information in the model training process, so as to better analyze and compare the time spent by each model in obtaining a closer recognition result, and evaluate the time performance of each model.

People's Daily 2014 Dataset. As shown in Table 8, the total training time required for TIPF is much lower than that of other models to obtain the same F1 value on the People's Daily 2014 dataset. Compared with the Lattice LSTM model, the total training time required by TIPF is only 39% of the total training time of the Lattice LSTM model. Compared with the Lex-BERT model, the total training time required by TIPF is only 18% of the total training time of the Lex-BERT model. Compared with the BiLSTM-CRF model, the total training time required by TIPF is only 22% of the total training time of the BiLSTM-CRF model.

At the same time, the Lattice LSTM model achieved the highest F1 value of 84.08 after 31 rounds of training on the People's Daily 2014 (22W) dataset, while the highest F1 value of this model on the People's Daily 2014 (2K) dataset is 81.10. This shows that the recognition accuracy of the Lattice LSTM model decreases when the size of the training set decreases. However, TIPF achieved the best results after only 22 rounds of training on the People's Daily 2014 (2K) than the Lattice LSTM model trained on the full dataset (People's Daily 2014 (22W)). In addition, the total training time required by TIPF in a low-performance environment is nearly 95,000 s less than the total training time of Lattice LSTM in a high-performance environment, which is only 17% of the total training time of the Lattice LSTM model.

Weibo NER Dataset. Table 9 shows that the total training time required for TIPF is much lower than that of other models. Compared with the Lattice LSTM and BiLSTM-CRF models, the total training time required for TIPF is only 69% and 56% of the total training time of the remaining two models. In addition, compared with the Lex-BERT

Table 8. Comparison of total training time on the 2014 dataset of People's Daily

Models	Type	F1	Time(s)
Lattice LSTM [10]	People's Daily 2014 (2K)	81.10	17454.96
Time for TIPF to obtain the same F1 value			6786.81
BiLSTM-CRF [20]	People's Daily 2014 (2K)	77.82	23631.20
Time for TIPF to obtain the same F1 value			5080.26
Lex-BERT [9]	People's Daily 2014 (2K)	82.48	53238.30
Time for TIPF to obtain the same F1 value			9344.35
Lattice LSTM [20]	People's Daily 2014 (22W)	84.08	114546.83
Time for TIPF to obtain the same F1 value			18843.28

model, the total training time of the TIPF is only 83% of the total training time of the Lex-BERT model, saving nearly 1100 s of training time.

Table 9. Comparison of total training time on Weibo NER dataset

Models	F1	Time(s)
Lattice LSTM [10]	41.39	4943.24
Time for TIPF to obtain the same F1 value		3216.61
BiLSTM-CRF [20]	41.05	6037.9
Time for TIPF to obtain the same F1 value		3400.13
Lex-BERT [9]	42.39	6195.84
Time for TIPF to obtain the same F1 value		5091.17

Boson Dataset. Table 10 shows that the total training time required for TIPF is much lower than that of other models. Compared with the Lattice LSTM model, the total training time required by TIPF is only 22% of the total training time of the Lattice LSTM model. Compared with the Lex-BERT model, the total training time required by TIPF is only 34% of the total training time of the Lex-BERT model. Compared with the BiLSTM-CRF model, the total training time required by TIPF is only 21% of the total training time of the BiLSTM-CRF model.

Table 10. Comparison of total training time on the Boson dataset

Models	F1	Time(s)
Lattice LSTM [10]	68.76	43585.95
Time for TIPF to obtain the same F1 value		9452.47
BiLSTM-CRF [20]	67.00	29017.68
Time for TIPF to obtain the same F1 value		6041.27
Lex-BERT [9]	68.91	27833.21
Time for TIPF to obtain the same F1 value		9452.47

4 Conclusion

In recent years, introducing lexical information into models has been a research topic for Chinese named entity recognition tasks. In this paper, a method based on textual information memory perception is proposed for Chinese named entity recognition. The method fully integrates the global and local features of the text, and uses the LSTM hidden state corresponding to each character/word as an extra knowledge source for named entity recognition. In this way, after training with fewer training samples, the method can achieve a high recognition accuracy and meet the needs of named entity recognition for Chinese text. This paper tests the performance of the proposed method on multiple public datasets. Experimental results show that the proposed method can achieve better performance than state-of-the-art named entity recognition methods on small datasets.

References

1. Pan, L.A., Yg, A., Fw, B.: Chinese named entity recognition: the state of the art. ScienceDirect **473**(1), 37–53 (2022)
2. Jiao, K.N., Li, X., Zhu, R.C.: Overview of Chinese domain named entity recognition. Comput. Eng. Appl. **57**(16), 1–15 (2021)
3. Lauriola, I., Lavelli, A., Aiolli, F.: An introduction to deep learning in natural language processing: models, techniques, and tools. Neurocomputing **470**(1), 443–456 (2021)
4. Li, J.Z.: An improved Chinese named entity recognition method with TB-LSTM-CRF. Comput. Sci. Appl. **11**(3), 720–728 (2021)
5. Jia, C., Shi, Y., Yang, Q.: Entity enhanced BERT pretraining for Chinese NER. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pp. 6384–6396. Association for Computational Linguistics (2020)
6. Jia, C., Zang, Y.: Multi-Cell compositional LSTM for NER domain adaptation. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 5906–5917. Association for Computational Linguistics, Stroudsburg (2020)
7. Hobley, E.: Iterative named entity recognition with conditional random fields. Appl. Sci. **12**(1), 330 (2022)
8. Liu, B.Y., Wu, D.Y., Liu, X.R.: Chinese named entity recognition incorporating global word boundary features. J. Chin. Inf. Process. **31**(2), 86–91 (2017)

9. Zhu, W., Cheung, D.: Lex-BERT: enhancing BERT based NER with lexicons. In: International Conference on Learning Representations, pp. 3–7. OpenReview.net, Virtual Event (2021)
10. Yue, Z, Jie, Y.: Chinese NER using lattice LSTM. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, pp. 1554–1564. Publisher, Melbourne (2018)
11. Kai, Z., Na, S.: Text mining and analysis of treatise on febrile diseases based on natural language processing. World J. Trad. Chin. Med. **6**(01), 73–79 (2020)
12. Song, Y., Shi, S., Li. J.: Directional skip-gram: explicitly distinguishing left and right context for word embeddings. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 175–180. Association for Computational Linguistics, New Orleans (2018)
13. Ratinov, L., Roth, D.: Design challenges and misconceptions in named entity recognition. In: Proceedings of the Thirteenth Conference on Computational Natural Language Learning, pp. 147–155. Association for Computational Linguistics, Boulder (2009)
14. Luo, Y., Xiao, F., Zhao, H.: Hierarchical contextualized representation for named entity recognition. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 5, pp. 8441–8448 (2020)
15. Lee, H., Jin, S., Chu, H.: Learning to remember patterns: pattern matching memory networks for traffic forecasting. In: International Conference on Learning Representations, pp. 78–85. OpenReview.net, Virtual Event (2021)
16. Jian, M.A., Zhang, T., Chen, Y.: New inverted index storage scheme for Chinese search engine. J. Comput. Appl. **33**(7), 2031–2036 (2013)
17. Hu, C., Cheng, J.: Named entity recognition based on character-level language models and attention mechanism. Int. Core J. Eng. **6**(1), 196–201 (2020)
18. Li, X., Wang, W., Hu, X.: Selective kernel networks. In: CVF Conference on Computer Vision and Pattern Recognition, pp. 128–130. IEEE, Long Beach (2019)
19. Wang, D.Z., Michelakis, E., Franklin, M.J.: Probabilistic declarative information extraction. In: 2010 IEEE 26th International Conference on Data Engineering, pp. 173–176. IEEE, Long Beach (2010)
20. Yin, J.Z., Luo, S.L., Wu, Z.T.: Chinese named entity recognition with character-level BLSTM and soft attention model. J. Beijing Inst. Technol. **103**(01), 63–74 (2020)