Neural Processing Letters (2022) 54:3425–3442
https://doi.org/10.1007/s11063-022-10768-y

TFM: A Triple Fusion Module for Integrating Lexicon Information in Chinese Named Entity Recognition

Haitao Liu[1] · Jihua Song[1] · Weiming Peng[1] · Jingbo Sun[1] · Xianwei Xin[1]

Accepted: 4 February 2022 / Published online: 22 April 2022

Abstract
Due to the characteristics of the Chinese writing system, character-based Chinese named entity recognition models ignore the word information in sentences, which harms their performance. Recently, many works try to alleviate the problem by integrating lexicon information into character-based models. These models, however, either simply concatenate word embeddings, or have complex structures which lead to low efficiency. Furthermore, word information is viewed as the only resource from lexicon, thus the value of lexicon is not fully explored. In this work, we observe another neglected information, i.e., character position in a word, which is beneficial for identifying character meanings. To fuse character, word and character position information, we modify the key-value memory network and propose a triple fusion module, termed as TFM. TFM is not limited to simple concatenation or suffers from complicated computation, compatibly working with the general sequence labeling model. Experimental evaluations show that our model has performance superiority. The F1-scores on Resume, Weibo and MSRA are 96.19%, 71.12% and 95.63% respectively.

Keywords Chinese named entity recognition · Lexicon information · Information fusion · Natural language processing

1 Introduction

Named Entity Recognition (NER) is a fundamental task in natural language processing (NLP), aiming at recognizing specific entities in unstructured texts, such as persons, countries and institutions. It is the upstream task of many NLP tasks which prevail currently, including question generation [7], event extraction [41] and knowledge graph [10]. NER has been regarded as a sequence labeling task in English and the BiLSTM-CRF architecture [19] is always taken as the model backbone. Popular models are based on words, since there are natural delimiters in English sentences [5, 24, 33, 47].

However, word-based NER models do not work well in Chinese NER. There are no natural delimiters in Chinese sentences, which means that sentences must be segmented

✉ Jihua Song
songjh@bnu.edu.cn

[1] School of Artificial Intelligence, Beijing Normal University, Beijing 100875, China

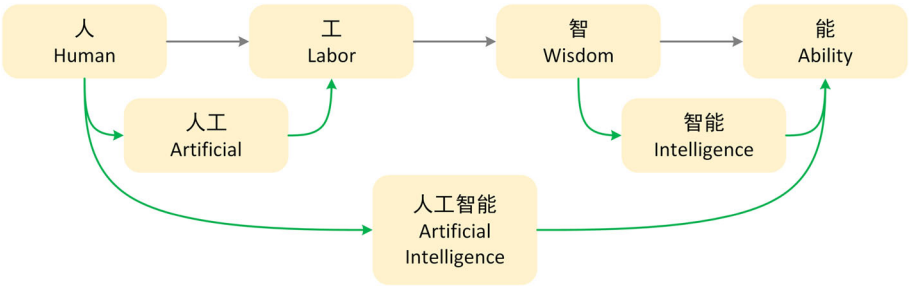

Fig. 1 The lattice structure. Green directed arrows denote information flow paths which connect word cells with their first character cells and last character cells

into words before a word-based model is applied. This is a tough process for machines and bad judgement may mislead NER inference. To eliminate the influence of word segmentation error, models based on characters have become mainstream in Chinese NER [9, 14, 37]. Later, Character-based models gradually hit a bottleneck because they ignore the fact that many Chinese characters express multiple meanings and only referring to words can most character meanings be determined. Guided by the idea that lexicon can play a supplementary role, researchers turn to integrating lexicon information into character-based models. Under this circumstance, the famous Lattice-LSTM [48] is proposed. The schematic diagram of Lattice-LSTM is shown in Fig. 1. More information flow paths are added to the standard LSTM layer [19], connecting words with their first characters and last characters. By a well-designed calculating method, it effectively integrates lexicon information into the character-based model. Though Lattice-SLTM is proven effective, its drawbacks are obvious. The complex model architecture greatly slows down its speed, which makes this model not very practical. Subsequent models like WC-LSTM [26], SoftLexicon [28] successfully solve the inefficiency problem of Lattice-LSTM, but they are limited in simply concatenating lexicon information and character embeddings. As a result, it still remains challenging to explore a more proper way to integrate lexicon information.

In addition, word information is often taken as the only information that can be obtained from lexicon in recent models. This information is presented in word embeddings while being integrated into models. According to our observation, however, another information can also be extracted from lexicon, i.e., character position in a word. This information is always not taken seriously by recent models, but sometimes it is valuable in distinguishing different meanings of a character. Take the character "代" as an example. On the one hand, if "代" is at the beginning position of a word, it most likely conveys the meaning of "substitute" or "acting" and in this condition character position information can help to recognize TITLE entities like "代理总经理" (acting general manager). On the other hand, if "代" is at the end of a word, it usually conveys the meaning of "era" or "epoch", which is conducive to identify TIME entities like "明代" (Ming Dynasty).

To utilize the character position information and solve the problems former models remain, we propose TFM, a triple fusion module which fuses character, word and character position information from lexicon. This module is inspired by the key-value memory network (KV-MemNN) [30], which is designed for the question answering task. The original KV-MemNN first incorporates key-value slots. Then it uses keys and a fixed question embedding to address weights, by which values are selectively retained. As shown in Fig. 2, the differences between KV-MemNN and TFM mainly reflect in the input form and the inner calculation process.

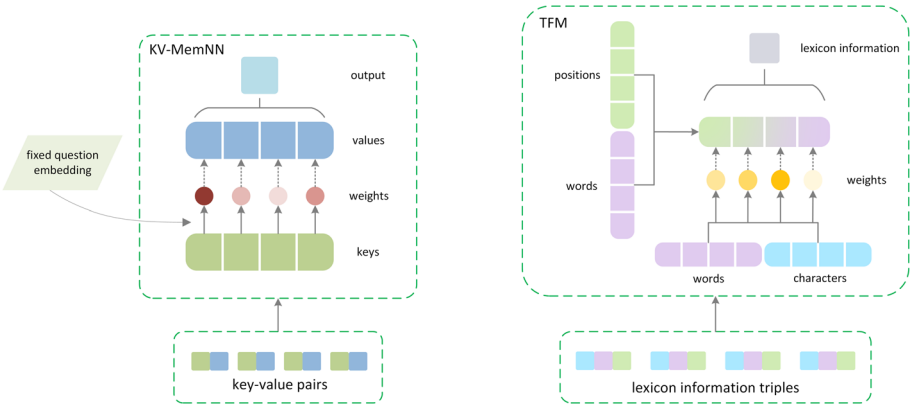

Fig. 2 Comparison between KV-MemNN and TFM. The KV-MemNN cell is on the left and the TFM cell is on the right

TFM takes lexicon information triples as input. Then weights are computed by the character and word information. Finally, the output is derived from the embeddings concatenated by the word and character position information. Though there are some differences, TFM keeps the main idea of KV-MemNN, i.e., memorizing information according to weights. Like a list of recent works [39, 40], we insert TFM into the general BiLSTM-CRF architecture to form our model. The contributions of our work can be summarized as follows:

1. We propose TFM to integrate lexicon information into the character-based Chinese NER model. TFM fuses information by neither complicated calculation nor simple concatenation, which is between these two.
2. Apart from word information, we exploit another information from lexicon which is overlooked by other methods, i.e., character position information. This information is fused together with the other two information in TFM and finally integrated into the character-based model.
3. To investigate the performance of our model, we evaluate it on three public Chinese NER datasets, i.e., Resume [48], Weibo [34] and MSRA [21]. By these experiments, we find that our model outperforms all the models for comparison.

The remaining sections of this article are organized as follows. Section 2 analyzes related work about this paper. Section 3 introduces the details of our model. Section 4 describes the setup of experiments and reports final results. Section 5 draws a conclusion.

2 Related Work

2.1 NER with Lexicon Information

The phenomenon of word information loss through lack of delimiters does not exist in languages like English. Therefore, there is no need to deliberately consider word information in NER in these languages. But utilizing words and phrases from lexicon can help models know specific entity instances in advance, thus enhancing their inference ability. For example, Liu et al. [25] concatenated query results to the output of BiLSTM and got tags by Semi-CRF. Peshterliev et al. [35] put gazetteer embeddings together with word embeddings, which means

they introduced lexicon information in the embedding layer. This concatenating method of fusing knowledge is also seen in Japanese and Hindi NER [13, 31]. Different from NER in word-based languages, the desire of word information in Chinese NER is not only confined to partial entity nouns, but also words in the whole sentences. So, the integration of word information needs further investigation.

Different methods have been proposed. On the one hand, jointly training and transfer learning worked. Peng and Dredze [34] jointly trained Chinese NER and Chinese word segmentation (CWS) models, improving the Chinese NER model with nearly 5% absolute improvement. Wu et al. [44] used CNN to capture local context and also jointly trained Chinese NER and CWS models. Cao et al. [3] applied adversarial transfer learning in Chinese NER, incorporating word boundary information from CWS task. On the other hand, lexicon information was valued. Ding et al. [8] constructed a directed acyclic graph to connect characters and words in lexicon, integrating both with a graph neural network. Gui et al. [15] and Sui et al. [37] utilized graph neural networks to integrate word information. Zhang and Yang [48] changed the architecture of standard LSTM, using shortcut paths to provide a link between character cells and word cells, which formed a lattice structure. In view of the complexity and inefficiency of lattice structure, Li et al. [23] proposed FLAT, Liu et al. [26] proposed WC-LSTM and Ma et al. [28] proposed SoftLexicon, all focusing on simplifying the complicated lattice structure, improving running speed and advancing the applicability of the models. Recently, Hu and Wei [18] rethought the second-order lexicon knowledge of the character which relieved word boundary conflicts. Gong et al. [12] constructed a hierarchical tree structure to utilize characters, subwords and lexicon words. Unlike all those models, we exploit one more information from lexicon, i.e., character position information and design TFM to integrate lexicon knowledge into the character-based model.

2.2 Key-Value Memory Network

KV-MemNN [30] was proposed for the question answering task, aiming at narrowing the gap between referring to knowledge bases and reading directly from documents. It encodes and integrates prior knowledge in key-value pairs. This method of incorporating information has a strong transferability and is often superior to simple embedding concatenation. As a result, it has been applied to other tasks, such as image recognition [2], clinical diagnostic inferencing [36] and machine translation [42]. In sequence labeling task, KV-MemNN has also proven its merits. Tian et al. [40] utilized wordhood information with KV-MemNN for CWS. In Slot Filling for dialogue systems, Wu et al. [45] adopted this method to trace long-term slot context. For NER task, to get document-level representation, Gui et al. [16] recorded context representations and label embeddings while Luo et al. [27] incorporated word representations and hidden states, all following the idea of KV-MemNN. Besides, some researchers leveraged syntactic information. Nie et al. [32] learned three syntactic information and Tian et al. [39] injected syntactic knowledge into the biomedical NER model. It is worth noting that all the mentioned methods for sequence labeling task generate various key-value pairs according to the task objectives and keep the basic structure of KV-MemNN unchanged. Different from these models, we incorporate lexicon knowledge into the NER model and modify KV-MemNN to make it suitable for the input of triples, instead of key-value pairs.

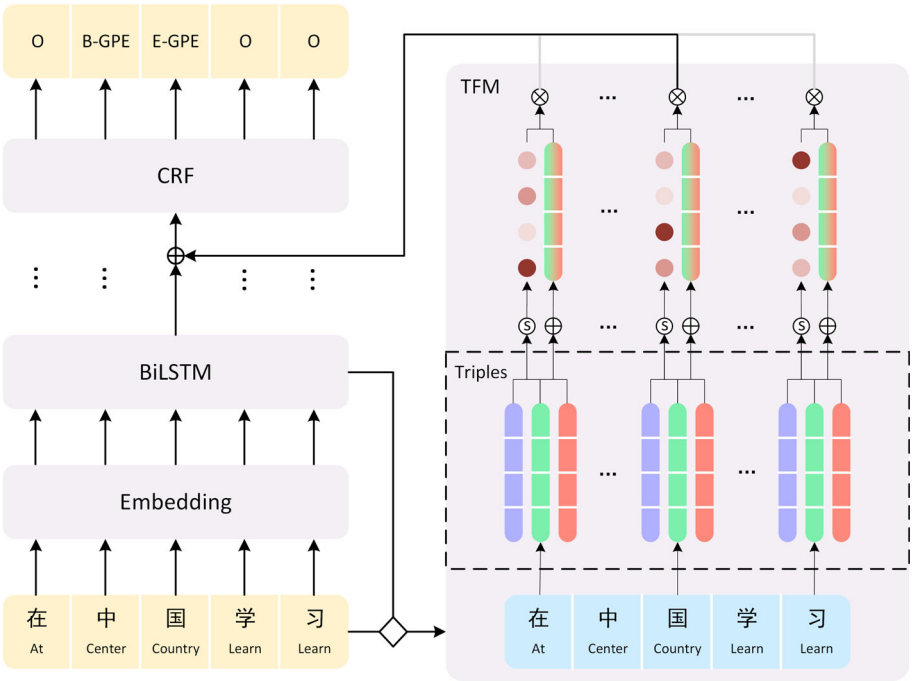

Fig. 3 The architecture of our model. The left part is the standard BiLSTM-CRF model and the right part is the proposed TFM. Word information is fused with character and character position information, where $\oplus$ denotes concatenation operation, $\otimes$ denotes element-wise product operation, $\textcircled{s}$ denotes the softmax function and $\diamond$ denotes the formation of triples

3 Method

In this work, we propose TFM to incorporate lexicon information into the character-based model. The architecture of our model is illustrated in Fig. 3, where the general BiLSTM-CRF model is on the left part with TFM on the right part working between the BiLSTM layer and the CRF layer. The first layer is an embedding layer which maps characters into dense vectors. The second layer is a BiLSTM layer by which character representations with context information are obtained. Then, triples containing character, word and character position information are sent to TFM to get fusion information. The last layer is the CRF layer whose input is the fusion information and output is the predicted named entity tags. Details are elaborated in the rest of this section.

3.1 Embedding Layer

In the embedding layer, characters in the input sentence in text form are mapped into values and expressed as dense vectors. Formally, given an input sentence $X = \{x_1, x_2, \ldots, x_n\} \in V_C$, where n denotes the length of the sentence and V_C denotes the character vocabulary, each character x_i is represented as:

$$\boldsymbol{e}_i^c = BERT\,(x_i)\,, \tag{1}$$

where $BERT(\cdot)$ denotes the pre-trained BERT [6] model. Recently, this model which integrates abundant semantic information has been widely used in NLP tasks. Rather than static embeddings, BERT can generate dynamic embeddings for the same character depending on its context characters.

3.2 BiLSTM Layer

BiLSTM [19] is good at capturing context information. Since the standard structure of BiLSTM has not been modified in our model, we briefly introduce its forward calculation process:

$$\begin{bmatrix} \boldsymbol{i}_i \\ \boldsymbol{f}_i \\ \boldsymbol{o}_i \\ \widetilde{\boldsymbol{c}}_i \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{bmatrix} \left(\boldsymbol{W} \begin{bmatrix} \boldsymbol{e}_i^c \\ \boldsymbol{h}_{i-1} \end{bmatrix} + \boldsymbol{b} \right), \tag{2}$$

$$\boldsymbol{c}_i = \widetilde{\boldsymbol{c}}_i * \boldsymbol{i}_i + \boldsymbol{c}_{i-1} * \boldsymbol{f}_i, \tag{3}$$

$$\boldsymbol{h}_i = \boldsymbol{o}_i * tanh(\boldsymbol{c}_i), \tag{4}$$

where σ is the sigmoid function, $*$ is element-wise product, $\boldsymbol{W}$ and $\boldsymbol{b}$ are trainable parameters.

Given a sequence of character embeddings, BiLSTM is applied to exploit hidden expressions of characters from global context:

$$\begin{aligned} \left[\overrightarrow{\boldsymbol{h}}_1, \overrightarrow{\boldsymbol{h}}_2, \ldots, \overrightarrow{\boldsymbol{h}}_n \right] &= \overrightarrow{LSTM} \left(\left[\boldsymbol{e}_1^c, \boldsymbol{e}_2^c, \ldots, \boldsymbol{e}_n^c \right] \right), \\ \left[\overleftarrow{\boldsymbol{h}}_1, \overleftarrow{\boldsymbol{h}}_2, \ldots, \overleftarrow{\boldsymbol{h}}_n \right] &= \overleftarrow{LSTM} \left(\left[\boldsymbol{e}_1^c, \boldsymbol{e}_2^c, \ldots, \boldsymbol{e}_n^c \right] \right), \end{aligned} \tag{5}$$

where $\overrightarrow{LSTM}$ and $\overleftarrow{LSTM}$ denote the forward and backward LSTMs and $\boldsymbol{e}_i^c (i = 1, 2, \ldots, n)$ are character embeddings. Finally, we get the representation with context information of x_i by concatenating $\overrightarrow{\boldsymbol{h}}_i$ and $\overleftarrow{\boldsymbol{h}}_i$:

$$\boldsymbol{h}_i = \left[\overrightarrow{\boldsymbol{h}}_i, \overleftarrow{\boldsymbol{h}}_i \right]. \tag{6}$$

3.3 Triple Fusion Module

To make the idea of KV-MemNN more suitable for our need of employing triples as the input, we innovatively change its architecture, fusing information in three steps.

Generating triples. For each character x_i in the input sentence X, we first get its matched words in a lexicon, represented as $W_i = \{w_{i1}, w_{i2}, \ldots, w_{im}\}$. Here, $w_{i\cdot}$ is a sub-sequence of X that contains x_i, i.e., $w_{i\cdot} = \{x_{i-a}, \ldots, x_i, \ldots, x_{i+b}\}$, where $0 \le a \le i$ and $0 \le b \le n-i$. For each word w_{ij} in W_i, a triple is generated by:

$$t_{ij} = \left(x_i, p_{ij}, w_{ij} \right), \tag{7}$$

which means x_i is at the p_{ij} position of w_{ij}. Here, p_{ij} is a position tag and it is an item of $\{B, E, S, M_1, M_2, \ldots, M_{k-2}\}$, where k is the maximum length of the words in the lexicon. Specifically, like common sequence labeling tags, B denotes that the character is at the **B**eginning position of the word, other labels can be deduced in the same manner. It should be noted that S is viewed as a position tag which denotes that the word consists of a single character. Furthermore, we distinguish the middle positions because we believe that different middle characters are closer to different parts in a word and the detailed character position information is good for NER inference, e.g., character "工" (labor) and "智" (intelligence)

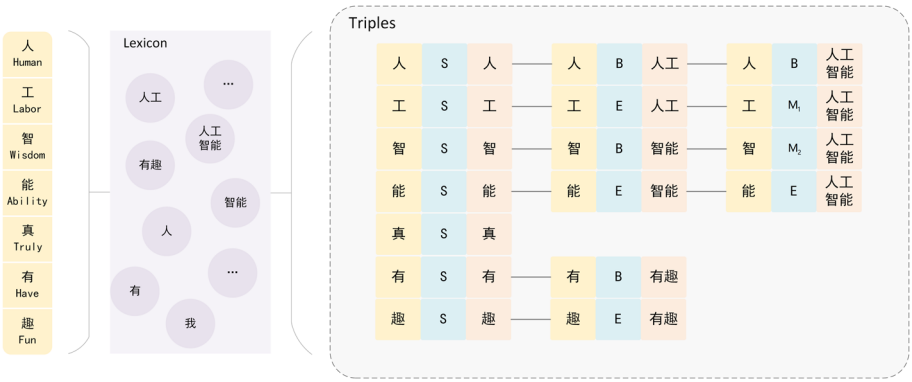

Fig. 4 An example of generating triples. The triples strung together with lines belong to the same character

are all at the middle positions in "人工智能" (artificial intelligence), but "工" (labor) is closer to "人" (human) while "智" (intelligence) is closer to "能" (ability) from lexicology. Consequently, the positions of "工" (labor) and "智"(intelligence) should not be confused. We use M_1 to denote the **first Middle** position of the word, M_2 denotes the **second Middle** position of the word and so on.

To show the process of generating triples concretely, we take "人工智能真有趣" (artificial intelligence is interesting) as an example. The character "工" (labor) occurs in three words, "工" (labor), "人工" (artificial) and "人工智能" (artificial intelligence). Then three triples are generated as $(工, S, 工)$, $(工, E, 人工)$ and $(工, M_1, 人工智能)$ and they mean that "工" (labor) is a separate word, "工" (labor) is at the end position of "人工" (artificial) and "工" (labor) is at the first middle position of "人工智能" (artificial intelligence). Triples belonging to "智" (intelligence) can be produced as $(智, S, 智)$, $(智, B, 智能)$, $(智, M_2, 人工智能)$ in the same way. The illustration is shown in Fig. 4.

*Digitizing triple sets.*The triples are digitalized before TFM fusing the information they contain. For each triple t_{ij} of x_i, the word embedding and character position embedding are mapped as follows:

$$\boldsymbol{e}_{ij}^{w} = Word\left(w_{ij}\right), \tag{8}$$

$$\boldsymbol{e}_{ij}^{p} = Position\left(p_{ij}\right), \tag{9}$$

where $Word(\cdot)$ and $Position(\cdot)$ are embedding lookup tables for words and character positions. Also, if the embedding of x_i in t_{ij} is represented as $\boldsymbol{e}_{ij}^{x}$, then the triple is updated as:

$$\boldsymbol{e}_{ij}^{t} = \left(\boldsymbol{e}_{ij}^{x}, \boldsymbol{e}_{ij}^{p}, \boldsymbol{e}_{ij}^{w}\right). \tag{10}$$

In this work, since $\boldsymbol{h}_i$ can be viewed as the character representation with context information, we set $\boldsymbol{e}_{ij}^{x}$ $(j = 1, \ldots, m)$ as $\boldsymbol{h}_i$.

Fusing information. After getting digitized triples, we fuse the character, word and character position information. Considering the instability of word frequency calculation caused by different corpora and following the main idea of KV-MemNN, for each word w_{ij} in the triple

set of x_i, its weight is computed by:

$$q_{ij} = \frac{exp\left(\boldsymbol{e}_{ij}^{x} \cdot \boldsymbol{e}_{ij}^{w}\right)}{\sum_{j=1}^{m} exp\left(\boldsymbol{e}_{ij}^{x} \cdot \boldsymbol{e}_{ij}^{w}\right)}. \tag{11}$$

Then word information and character position information are concatenated:

$$\boldsymbol{e}_{ij}^{f} = \boldsymbol{W}_f \cdot \boldsymbol{e}_{ij}^{w} \oplus \boldsymbol{e}_{ij}^{p}, \tag{12}$$

where $\boldsymbol{W}_f$ is a trainable parameter. The lexicon information of x_i will be computed by:

$$\boldsymbol{e}_{i}^{l} = \sum_{j=1}^{m} q_{ij} \boldsymbol{e}_{ij}^{f}. \tag{13}$$

Afterwards, $\boldsymbol{e}_{i}^{l}$ and $\boldsymbol{h}_i$ are concatenated and returned as the fusion information:

$$\boldsymbol{v}_i = \boldsymbol{h}_i \oplus \boldsymbol{e}_{i}^{l}. \tag{14}$$

Here, we still take the former example to describe the details. As is shown in the right part of Fig. 2, the triples assigned to the three words of character x_2 are $\left(\boldsymbol{e}_{21}^{x}, \boldsymbol{e}_{21}^{p}, \boldsymbol{e}_{21}^{w}\right)$, $\left(\boldsymbol{e}_{22}^{x}, \boldsymbol{e}_{22}^{p}, \boldsymbol{e}_{22}^{w}\right)$, $\left(\boldsymbol{e}_{23}^{x}, \boldsymbol{e}_{23}^{p}, \boldsymbol{e}_{23}^{w}\right)$. In this step, weights q_{21}, q_{22}, q_{23} are first calculated by Eq. 11. Then the word and character position embeddings are concatenated as $\boldsymbol{e}_{21}^{f}, \boldsymbol{e}_{22}^{f}, \boldsymbol{e}_{23}^{f}$. Based on the weights and the concatenating embeddings, the lexicon information $\boldsymbol{e}_{2}^{l}$ is calculated according to Eq. 13. Finally, fusion information $\boldsymbol{v}_2$ is output by putting $\boldsymbol{e}_{2}^{l}$ together with $\boldsymbol{h}_2$.

3.4 CRF Layer

A standard CRF [20] layer is used at the top of BiLSTM layer and TFM. Given the predicted tag sequence $Y = \{y_1, y_2, \ldots, y_n\} \in V_l$, where V_l denotes the label set, the probability of the predicted sequence is

$$P\left(Y \mid X\right) = \frac{exp\left(\sum_i \left(\boldsymbol{W}_{CRF}^{y_i} \boldsymbol{v}_i + \boldsymbol{b}_{CRF}^{(y_{i-1}, y_i)}\right)\right)}{\sum_{Y'} exp\left(\sum_i \left(\boldsymbol{W}_{CRF}^{y'_i} \boldsymbol{v}_i + \boldsymbol{b}_{CRF}^{(y'_{i-1}, y'_i)}\right)\right)}, \tag{15}$$

where Y' denotes an arbitrary tag sequence, and $\boldsymbol{W}_{CRF}^{y_i}$ and $\boldsymbol{b}_{CRF}^{(y_{i-1},\ y_i)}$ are trainable parameters. We use Viterbi algorithm [11] to get the predicted tag sequence. Given a set of training data $\{(X_i, Y_i)\}|_{i=1}^{N}$, a log-likelihood loss function is used to train the model:

$$L = \sum_{i=1}^{N} \log P\left(Y_i | X_i\right). \tag{16}$$

4 Experiments

We conduct a series of experiments on public Chinese NER datasets to study the effectiveness of our model. Standard precision (P), recall (R) and F1-score (F1) are used to evaluate the performance.

Table 1 Hyper-parameter settings

Parameter	Value
Word emb size	50
LSTM hidden size	300
Dropout	0.5
TFM dropout	0.5
Learning rate	0.0075
Learning rate decay	0.05
LSTM layer	1
Optimizer	adam

4.1 Experiment Setup

Preparation. We perform experiments on three public Chinese NER datasets: Resume [48], Weibo [34] and MSRA [21]. Three datasets are collected from different domains. Specifically, Resume is collected from Sina Finance,[1] Weibo is collected from Sina Weibo[2] and MSRA is collected from newswire. Gold segmentation is not available for the mentioned datasets. The lexicon we used is released by [48] and contains 704.4k words. The position lookup table is randomly initialized and trained. The BERT pre-trained model we use is bert-base-chinese[3] and its parameters are fixed. The tagging styles are all transformed to BMES tagging style for the unity of experiments.

Hyper-parameter settings. The main hyper-parameters we set during experiments are shown in Table 1. We keep some parameters the same with Lattice-LSTM, including word embedding size, dropout, learning rate decay and so on. Other parameters like LSTM hidden size and learning rate are adjusted to fit our model.

Models for comparison. Since our model focuses on integrating lexicon information, recent models which share the same goal with our model are selected as the baselines, i.e.,

1. **BERT-tagger** [6] fine-tunes the BERT for encoding and uses a classification layer for decoding.
2. **BERT+BiLSTM-CRF** is based on BiLSTM-CRF and uses BERT as the encoder.
3. **CAN-NER** [49] captures the context information with a character-based CNN and a gated GRU.
4. **self-attention+BiLSTM-CRF** [4] adds the self-attention mechanism to the BiLSTM-CRF model, which integrates character and word information.
5. **BERT+MFE** [22] incorporates semantic, glyph and phonetic features into the character-based model.
6. **Multi-digraph model** [8] uses graph neural networks to integrate lexicon information.
7. **LGN** [15] utilizes a graph neural network to solve word ambiguities by integrating characters, potential words and sentence semantics.
8. **CGN** [37] is proposed to integrate self-matched lexical words and nearest contextual lexical words.
9. **Lattice-LSTM** [48] is a variant of LSTM which incorporates all potential words into a character-based model.

[1] http://finance.sina.com.cn/stock/index.shtml.

[2] http://www.weibo.com/.

[3] https://huggingface.co/bert-base-chinese.

10. **LR-CNN** [14] is a CNN-based method that incorporates lexicons using a rethinking mechanism.
11. **WC-LSTM** [26] adds word information to the start or the end character of the word, aiming at solving some problems that have been found in Lattice-LSTM. In WC-LSTM, there are four words encoding strategy. For each dataset, we pick the strategy which works best for comparison.
12. **HiLSTM** [12] is a hierarchical LSTM framework which considers not only words in lexicon but also words and subwords in sentences.
13. **BERT+SoftLexicon** [28] follows the idea of Lattice-LSTM, which avoids the complex structure and improves the performance of model.
14. **SLK-NER** [18] fuses different second-order lexicon knowledge (SLK) with the global attention information to alleviate the impact of word boundary conflicts.
15. **AM-BiLSTM** [46] enhances character embeddings with the multi-word information feature, which keeps the word information by matching a lexicon.

4.2 Results on Benchmark Datasets

Experimental results on Resume, Weibo and MSRA are respectively shown in Tables 2, 3, 4. We divide the baselines into four groups, including general models in Chinese NER which do not integrate any lexicon information, models using different external knowledge, models utilizing lexicon information by graph neural network and the state-of-the-art Chinese NER models which focus on integrating lexicon information.

As we can see from the tables, compared with baselines in the first group, our model outperforms all the models. This proves that integrating lexicon information has a positive effect on the character-based model.

Table 2 Results on resume

Models	P	R	F1
BERT-tagger [28]	94.87	**96.50**	95.68
BERT+BiLSTM-CRF [28]	95.75	95.28	95.51
CAN-NER [49]	95.05	94.82	94.94
Self-attention+BiLSTM-CRF [4]	–	–	–
BERT+MFE [22]	95.76	95.71	95.73
Multi-digraph model [8]	–	–	–
LGN [15]	95.28	95.46	95.37
CGN [37]	–	–	–
Lattice-LSTM [48]	94.81	94.11	94.46
LR-CNN [14]	95.37	94.84	95.11
WC-LSTM+longest [26]	95.27	95.15	95.21
HiLSTM [12]	–	–	–
BERT+SoftLexicon [28]	96.08	96.13	96.11
SLK-NER [18]	95.20	96.40	95.80
AM-BiLSTM [46]	94.29	95.63	94.95
Our model	**96.37**	96.01	**96.19**

Table 3 Results on Weibo

Models	NE	NM	All
BERT-tagger [28]	65.77	62.05	63.80
BERT+BiLSTM-CRF [28]	69.65	64.62	67.33
CAN-NER [49]	55.38	62.98	59.31
Self-attention+BiLSTM-CRF [4]	–	–	61.46
BERT+MFE [22]	–	–	67.74
Multi-digraph model [8]	–	–	59.50
LGN [15]	55.34	64.98	60.21
CGN [37]	56.45	68.32	63.09
Lattice-LSTM [48]	53.04	62.25	58.79
LR-CNN [14]	57.14	66.67	59.92
WC-LSTM+longest [26]	52.55	67.41	59.84
HiLSTM [12]	60.94	**68.89**	63.79
BERT+SoftLexicon [28]	70.94	67.02	70.50
SLK-NER [18]	–	–	64.00
AM-BiLSTM [46]	–	-	–
Our model	**71.29**	67.04	**71.12**

NE, NM and All denote F1-scores for named entities, nominal entities and both

Table 4 Results on MSRA

Models	P	R	F1
BERT-tagger [28]	93.40	94.12	93.76
BERT+BiLSTM-CRF [28]	95.06	94.61	94.83
CAN-NER [49]	93.53	92.42	92.97
Self-attention+BiLSTM-CRF [4]	95.92	94.80	95.36
BERT+MFE [22]	93.32	86.83	89.96
Multi-digraph model [8]	94.60	94.20	94.40
LGN [15]	94.19	92.73	93.46
CGN [37]	94.01	92.93	93.47
Lattice-LSTM [48]	93.57	92.79	93.18
LR-CNN [14]	94.50	92.93	93.71
WC-LSTM+average [26]	94.58	92.91	93.74
HiLSTM [12]	94.83	93.61	94.22
BERT+SoftLexicon [28]	95.75	**95.10**	95.42
SLK-NER [18]	–	–	–
AM-BiLSTM [46]	–	–	–
Our model	**96.29**	94.97	**95.63**

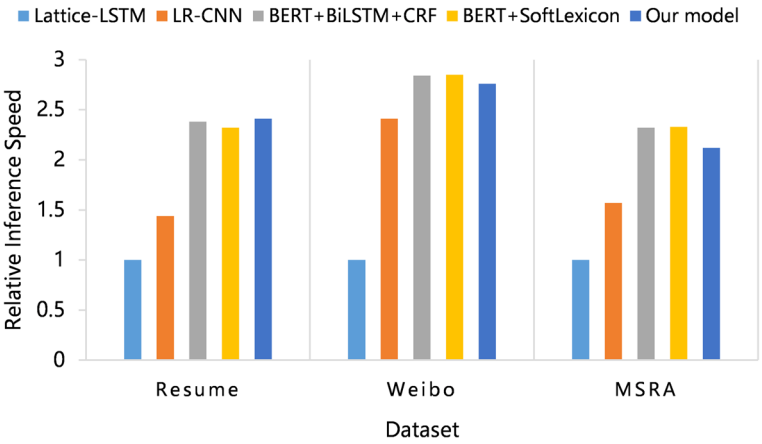

Fig. 5 Relative inference speed on three datasets

Baselines in the second group integrate different knowledge, including word segmentation information and character feature information. Our model performs better than them, since there are always some mistakes during word segmentation and character features cannot provide enough information for NER inference.

When we compare our model with baselines in the third group, we find that the F1-scores of graph-based models are not better than our model. One of the reasons is that these baselines do not exploit full lexicon information. Our TFM overcomes this shortcoming very well.

Our model outperforms all the state-of-the-art baselines in the fourth group. We attribute the improvements to the TFM we proposed because TFM integrates more lexicon information and the multiple information are fused in a more proper manner.

In all, our model achieves the best F1 and P, which proves its merits.

4.3 Efficiency Study

In this section, we evaluate the inference speed of our model. The settings follow [28]. We choose the general BERT+BiLSTM-CRF and three models which integrate lexicon information for comparison.

Results are shown in Fig. 5. The inference speed of our model is faster than Lattice-LSTM and LR-CNN and is more than twice Lattice-LSTM. This proves the improvement of efficiency. Though TFM is inserted, the inference speed of our model is very close to the BERT+BiLSTM-CRF model. Moreover, SoftLexicon focuses on integrating lexicon information while reducing running time. Our model integrates more information, but the inference speed does not significantly drop.

4.4 Influence of Different Sequence Modeling Layers

Though general sequence labeling model BiLSTM-CRF is used as the backbone of our model, we still want to explore the influence of different sequence modeling layers. To this end, we replace BiLSTM with CNN and transformer[43]. The F1-socres are shown in Table 5.

Table 5 F1-scores with different sequence modeling layers

Layer	Resume	Weibo	MSRA
CNN	94.41	67.10	94.87
Transformer	94.76	67.34	95.01
BiLSTM	96.19	71.12	95.63

Table 6 F1-scores with different embedding methods

Embedding	Resume	Weibo	MSRA
Fasttext	93.25	56.76	91.80
word2vec	94.00	57.66	92.12
ERNIE	95.83	70.07	95.62
BERT	96.19	71.12	95.63

From the table, we find that our model empirically works best with BiLSTM and F1-socres drop when applying other sequence modeling layers. Since most models choose BiLSTM, we also follow them to make comparisons fairer.

4.5 Influence of Different Embedding Methods

We try three other embedding methods, i.e., fastText [1], word2vec [29] and ERNIE [38] to evaluate the influence of different embedding methods. For fastText, we use the pre-trained model provided by Facebook.[4] For word2vec, we use the pre-trained model released by [48]. For ERNIE, we use a PyTorch version[5]. The results are shown in Table 6.

The F1-socres on fastText and word2vec are lower than BERT and ERNIE, especially on Weibo dataset. One of the reasons is that Weibo is in informal language environments and static embeddings cannot express exact meanings of characters. F1-scores with ERNIE are close to BERT, but BERT is used more widely and our model works better with BERT, so we choose BERT as our embedding layer.

4.6 Performance on Different Entities

We further analyze F1-scores of different entities to investigate whether the improvement of our model is general or entity-specific. We compare our model with Lattice-LSTM and BERT+BiLSTM-CRF. Experimental results are shown in Fig. 6.

The three models perform well on Resume and our model only improves on certain entity types, such as ORG, TITLE and PRO. On Weibo, the F1-scores of our model on four entity types exceed Lattice-LSTM. When compared with BERT+BiLSTM-CRF, we find that the F1-score of our model on ORG entity drops. The drop, however, does not affect the overall improvement, since the F1-scores of other entity types rise. Finally, our model has slight F1-score increases on three entity types in MSRA and this contributes to the overall advantages.

[4] https://fasttext.cc/docs/en/crawl-vectors.html.

[5] https://github.com/nghuyong/ERNIE-Pytorch.

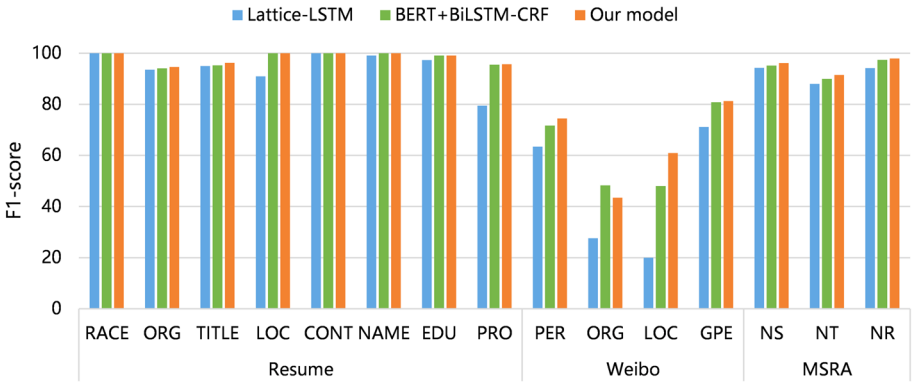

Fig. 6 F1-scores of different entity types on three datasets

Table 7 An example from Weibo

Sentence	亚马逊官方微信包裹跟踪服务上线 Amazon's official WeChat package tracking service launched														
Matched words	亚马逊，亚马，马逊，官方，微信，信包，包裹，跟踪，服务，上线 Amazon，Ya Ma，Ma Xun，Official，WeChat，Pack，Package，Track，Service，Launch														
Gold labels	亚	马	逊	官	方	微	信	包	裹	跟	踪	服	务	上	线
	B-ORG	M-ORG	E-ORG	O	O	O	O	O	O	O	O	O	O	O	O
Lattice-LSTM	亚	马	逊	官	方	微	信	包	裹	跟	踪	服	务	上	线
	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
Our model	亚	马	逊	官	方	微	信	包	裹	跟	踪	服	务	上	线
	B-ORG	M-ORG	E-ORG	O	O	O	O	O	O	O	O	O	O	O	O

The suffixes ". NAM" in the entity labels and words consist of single character are omitted for brevity. "Matched words" denotes the matched words from lexicon

4.7 Case Study

In this section, we analyze a sentence from Weibo dataset, i.e., "亚马逊官方微信包裹跟踪服务上线" (Amazon's official WeChat package tracking service launched). The design of this experiment is following [26]. Experimental results are shown in Table 7. Even with the help of lexicon knowledge, Lattice-LSTM model fails to recognize the organization entity "亚马逊" (Amazon). Different from Lattice-LSTM, our model correctly makes the predictions. Later, we check the lexicon for experiment and find that almost every word that ends with the character"逊" is an entity, e.g., "鲁滨逊" (Robinson), "陶逊" (Tao Xun) and "纳尔逊" (Nelson). In this situation, the position information of "逊" in the word plays an important role in recognizing named entity. The TFM fuses the character position information of "逊" in "亚马 逊" (Amazon) which contributes to the right inference of our model.

During the lexicon matching process, many words can be assigned to a character. In fact, most of the words are useless. Though all matched words are fused, we still hope that TFM pays more attention to latent right words. We record the weights of the words for each character in TFM and get the heatmap shown in Fig. 7. As we can see from the heatmap, right words like "亚马逊" (Amazon), "官方" (official) are assigned greater weights. This proves that TFM can automatically diminish the disturbance of useless words.

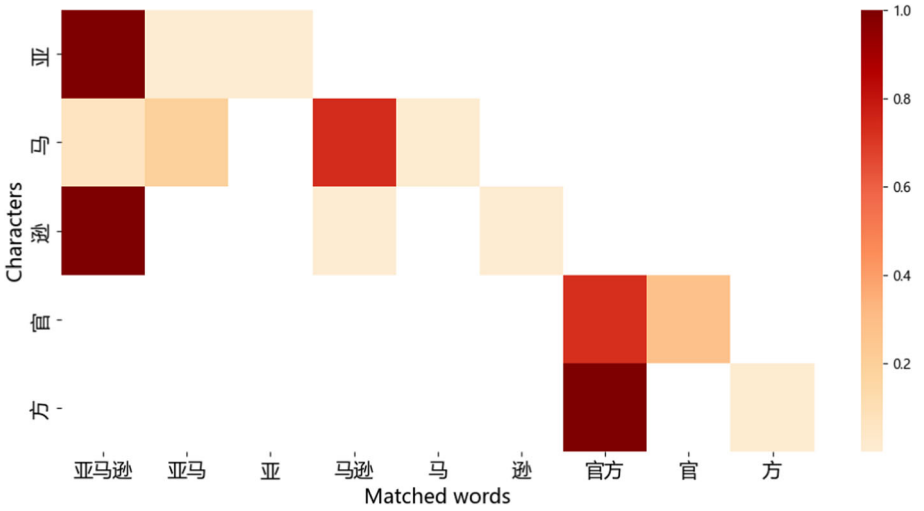

Fig. 7 Heatmap of the word weights for the example "亚马逊官方微信包裹跟踪服务上线" (Amazon's official WeChat package tracking service launched). We truncate the sentence for brevity. The darker the color, the greater the weight

Table 8 An ablation study on our model

Models	Resume	Weibo	MSRA
Our model	96.19	71.12	95.63
- TFM	95.51	67.33	94.83
- Position	95.97	69.59	94.56
- Word	95.65	68.97	94.54
- "M" distinction	95.89	69.10	94.90

"-TFM" denotes that TFM is not included. "-position" denotes that TFM does not fuse character position information. "-word" denotes that TFM does not fuse word information. "- 'M' distinction" denotes that middle positions are not distinguished in TFM

4.8 Ablation Study

To investigate different factors that affect the performance of our model, we conduct ablation study on three datasets and the results are shown in Table 8.

1. In the "- TFM" experiment, we remove the proposed TFM. In this case, the model turns to the BERT+BiLSTM-CRF model. We find that the performance of the model drops, which demonstrates the effectiveness of our proposed module.
2. By "-position" and "-word" experiment, we study the influence of different information combinations. When only one kind of information from lexicon is fused, the F1-scores drop. This proves that our model works best when combining the word and character position information.
3. Affected by the labeling paradigm in sequence labeling, models like SoftLexicon overlook the distinction of different middle positions and put them in the same group. In the "- 'M' distinction" experiment, we do not distinguish any middle positions and the performance

of model declines. This proves that emphasizing different middle positions during fusing process has a positive effect on the performance of our model.

5 Conclusion

In this paper, we notice the character position information from lexicon which is ignored by other models. Then TFM is proposed to integrate lexicon knowledge into the character-based model. TFM effectively handles the situation that word information is lost in the character-based model and enhances the ability to understand the relationship between a word and its characters by the character position information. Experiments on public datasets show that our model achieves ideal performance in terms of efficiency and F1-score.

In the end, we describe our future work. The model we propose works well when the training data is sufficient. But it will overfit under the circumstances of extremely scarce training examples, i.e., few-shot settings [17]. So, we plan to explore how to integrate external knowledge into few-shot NER models.

Acknowledgements This work was supported by the National Natural Science Foundation of China (Grant Nos. 61877004 and 62007004), the Major Program of National Social Science Foundation of China (Grant No. 18ZDA295) and the Doctoral Interdisciplinary Foundation Project of Beijing Normal University (Grant No. BNUXKJC2020).

References

1. Bojanowski P, Grave E, Joulin A, Mikolov T (2017) Enriching word vectors with subword information. Trans Assoc Comput Linguist 5:135–146
2. Cai Q, Pan Y, Yao T, Yan C, Mei T (2018) Memory matching networks for one-shot image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4080–4088
3. Cao P, Chen Y, Liu K, Zhao J, Liu S (2018) Adversarial transfer learning for chinese named entity recognition with self-attention mechanism. In: Proceedings of the 2018 conference on empirical methods in natural language processing, pp 182–192
4. Chang N, Zhong J, Li Q, Zhu J (2020) A mixed semantic features model for Chinese NER with characters and words. Adv Inf Retr 12035:356
5. Chiu JP, Nichols E (2016) Named entity recognition with bidirectional LSTM-CNNS. Trans Assoc Comput Linguist 4:357–370
6. Devlin J, Chang MW, Lee K, Toutanova K (2018) Bert: pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805
7. Dhole KD, Manning CD (2020) Syn-qg: syntactic and shallow semantic rules for question generation. arXiv:2004.08694
8. Ding R, Xie P, Zhang X, Lu W, Li L, Si L (2019) A neural multi-digraph model for chinese ner with gazetteers. In: Proceedings of the 57th annual meeting of the association for computational linguistics, pp 1462–1467
9. Dong C, Zhang J, Zong C, Hattori M, Di H (2016) Character-based LSTM-CRF with radical-level features for Chinese named entity recognition. In: Natural language understanding and intelligent applications. Springer, pp 239–250
10. Elhammadi S, Lakshmanan LV, Ng R, Simpson M, Huai B, Wang Z, Wang L (2020) A high precision pipeline for financial knowledge graph construction. In: Proceedings of the 28th international conference on computational linguistics, pp 967–977
11. Forney GD (1973) The viterbi algorithm. Proc IEEE 61(3):268–278
12. Gong C, Li Z, Xia Q, Chen W, Zhang M (2020) Hierarchical LSTM with char-subword-word tree-structure representation for Chinese named entity recognition. Sci China Inf Sci 63(10):1–15
13. Goyal A, Gupta V, Kumar M (2021) A deep learning-based bilingual hindi and punjabi named entity recognition system using enhanced word embeddings. Knowl Based Syst, 107601

14. Gui T, Ma R, Zhang Q, Zhao L, Jiang YG, Huang X (2019) CNN-based Chinese ner with lexicon rethinking. In: IJCAI, pp 4982–4988
15. Gui T, Zou Y, Zhang Q, Peng M, Fu J, Wei Z, Huang XJ (2019) 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), pp 1039–1049
16. Gui T, Ye J, Zhang Q, Zhou Y, Gong Y, Huang X (2020) Leveraging document-level label consistency for named entity recognition. In: IJCAI, pp 3976–3982
17. Hofer M, Kormilitzin A, Goldberg P, Nevado-Holgado A (2018) Few-shot learning for named entity recognition in medical text. arXiv:1811.05468
18. Hu D, Wei L (2020) SLK-NER: exploiting second-order lexicon knowledge for Chinese NER. arXiv:2007.08416
19. Huang Z, Xu W, Yu K (2015) Bidirectional LSTM-CRF models for sequence tagging. arXiv:1508.01991
20. Lafferty J, McCallum A, Pereira FC (2001) Conditional random fields: probabilistic models for segmenting and labeling sequence data
21. Levow GA (2006) The third international chinese language processing bakeoff: Word segmentation and named entity recognition. In: Proceedings of the Fifth SIGHAN workshop on Chinese language processing, pp 108–117
22. Li J, Meng K (2021) MFE-NER: multi-feature fusion embedding for chinese named entity recognition. arXiv:2109.07877
23. Li X, Yan H, Qiu X, Huang X (2020) Flat: Chinese NER using flat-lattice transformer. arXiv:2004.11795
24. Lin BY, Lee DH, Shen M, Moreno R, Huang X, Shiralkar P, Ren X (2020) Triggerner: learning with entity triggers as explanations for named entity recognition. arXiv:2004.07493
25. Liu T, Yao JG, Lin CY (2019) Towards improving neural named entity recognition with gazetteers. In: Proceedings of the 57th annual meeting of the association for computational linguistics, pp 5301–5307
26. Liu W, Xu T, Xu Q, Song J, Zu Y (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, vol. 1 (Long and Short Papers), pp 2379–2389
27. Luo Y, Xiao F, Zhao H (2020) Hierarchical contextualized representation for named entity recognition. In: Proceedings of the AAAI conference on artificial intelligence, vol 34, pp 8441–8448
28. Ma R, Peng M, Zhang Q, Huang X (2019) Simplify the usage of lexicon in Chinese NER. arXiv:1908.05969
29. Mikolov T, Sutskever I, Chen K, Corrado G, Dean J (2013) Distributed representations of words and phrases and their compositionality. arXiv:1310.4546
30. Miller A, Fisch A, Dodge J, Karimi AH, Bordes A, Weston J (2016) Key-value memory networks for directly reading documents. arXiv:1606.03126
31. Misawa S, Taniguchi M, Miura Y, Ohkuma T (2017) Character-based bidirectional lstm-crf with words and characters for japanese named entity recognition. In: Proceedings of the first workshop on subword and character level models in NLP, pp 97–102
32. Nie Y, Tian Y, Song Y, Ao X, Wan X (2020) Improving named entity recognition with attentive ensemble of syntactic information. arXiv:2010.15466
33. Nie Y, Tian Y, Wan X, Song Y, Dai B (2020) Named entity recognition for social media texts with semantic augmentation. arXiv:2010.15458
34. Peng N, Dredze M (2016) Improving named entity recognition for Chinese social media with word segmentation representation learning. arXiv:1603.00786
35. Peshterliev S, Dupuy C, Kiss I (2020) Self-attention gazetteer embeddings for named-entity recognition. arXiv:2004.04060
36. Prakash A, Zhao S, Hasan SA, Datla V, Lee K, Qadir A, Liu J, Farri O (2017) Condensed memory networks for clinical diagnostic inferencing. In: Thirty-first AAAI conference on artificial intelligence
37. Sui D, Chen Y, Liu K, Zhao J, Liu S (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), pp 3821–3831
38. Sun Y, Wang S, Li Y, Feng S, Chen X, Zhang H, Tian X, Zhu D, Tian H, Wu H (2019) Ernie: enhanced representation through knowledge integration. arXiv:1904.09223
39. Tian Y, Shen W, Song Y, Xia F, He M, Li K (2020) Improving biomedical named entity recognition with syntactic information. BMC Bioinform 21(1):1–17
40. Tian Y, Song Y, Xia F, Zhang T, Wang Y (2020) Improving chinese word segmentation with wordhood memory networks. In: Proceedings of the 58th annual meeting of the association for computational linguistics, pp 8274–8285

41. Tong M, Xu B, Wang S, Cao Y, Hou L, Li J, Xie J (2020) Improving event detection via open-domain trigger knowledge. In: Proceedings of the 58th annual meeting of the association for computational linguistics, pp 5887–5897
42. Tu Z, Liu Y, Shi S, Zhang T (2018) Learning to remember translation history with a continuous cache. Trans Assoc Comput Linguist 6:407–420
43. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017) Attention is all you need. arXiv:1706.03762
44. Wu F, Liu J, Wu C, Huang Y, Xie X (2019) Neural Chinese named entity recognition via CNN-LSTM-CRF and joint training with word segmentation. In: The World Wide Web conference, pp 3342–3348
45. Wu J, Harris I, Zhao H (2021) Spoken language understanding for task-oriented dialogue systems with augmented memory networks. In: Proceedings of the 2021 conference of the North American chapter of the association for computational linguistics: human language technologies, pp 797–806
46. Xu H, Chen Z, Wang S, Jiang X (2021) Chinese NER using Albert and multi-word information. In: ACM turing award celebration conference-China (ACM TURC 2021), pp 141–145
47. Yan R, Jiang X, Dang D (2021) Named entity recognition by using XLNet-BILSTM-CRF. Neural Process Lett 53:1–18
48. Zhang Y, Yang J (2018) Chinese Ner using lattice LSTM. arXiv:1805.02023
49. Zhu Y, Wang G, Karlsson BF (2019) Can-ner: convolutional attention network for Chinese named entity recognition. arXiv:1904.02141