ALFLAT: Chinese NER Using ALBERT, Flat-Lattice Transformer, Word Segmentation and Entity Dictionary

Haifeng Lv[1,3](✉) and Yong Ding[2]

[1] School of Data Science and Software Engineering, Wuzhou University, Wuzhou, China
421538806@qq.com
[2] Guangxi Key Laboratory of Cryptography and Information Security, School of Computer Science and Information Security, Guilin University of Electronic Technology, Guilin, China
[3] Guangxi Key Laboratory of Machine Vision and Intelligent Control, Wuzhou University, Wuzhou, China

Abstract. Recently, the character-word lattice structure has been proved to be effective for Chinese named entity recognition (NER) by incorporating the word information. However, one hand, since the lattice structure is dynamic and complex, although some existing lattice-based models are effectively utilize the parallel computation of GPUs, they do not fully utilize word segmentation boundary tags that as features are helpful for NER task. On the other hand, the character-word vector needs to be trained, and the user-defined entity dictionary cannot be effectively used. In this paper, we propose ALFLAT: based on a flat-lattice Transformer to incorporate ALBERT pre-trained model, word segmentation information and user-defined entity dictionary for Chinese NER. ALFLAT converts the lattice structure into a flat structure consisting of spans, integrate word segmentation embedding with the output of flat-lattice Transformer model, then modifies the emission scores according to the user-defined entity dictionary, finally utilize Viterbi decoding of the CRF layer to obtain the correct entity results. Each span corresponds to a character or latent word and its position in the original lattice. With the power of ALBERT pre-trained model, Transformer and position encoding, ALFLAT can fully leverage the lattice, word segmentation and user-defined entity dictionary information. Experiments on MSRA dataset show ALFLAT outperforms other lexicon-based models in performance and efficiency.

Keywords: NER · ALBERT · Lattice transformer · CRF · Word segmentation

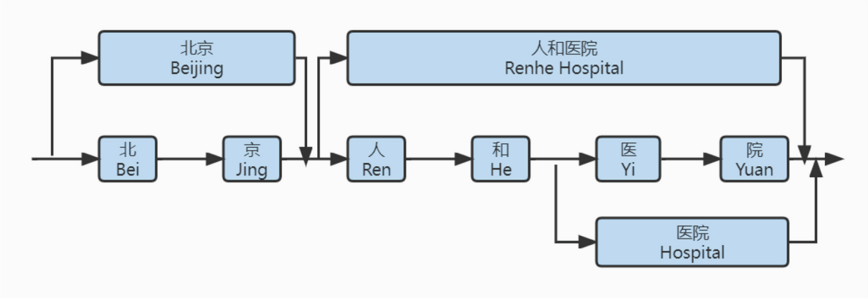
1 Introduction

Named entity recognition (NER) plays an indispensable role in many downstream natural language processing (NLP) tasks [1, 2]. Compared with English NER [3–6], Chinese NER is more difficult since it usually involves word segmentation.

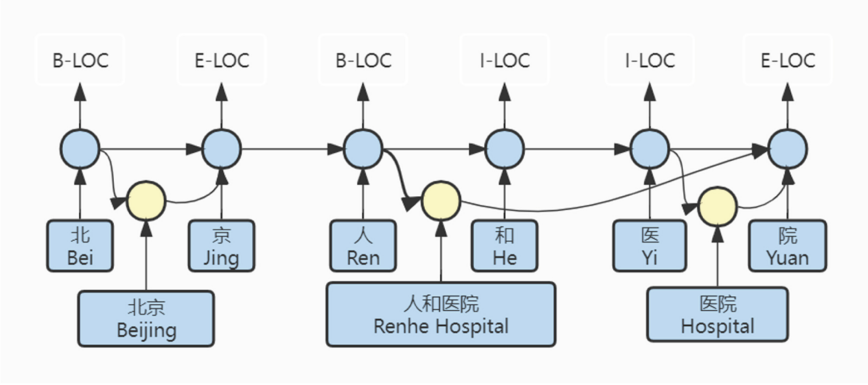
Recently, the lattice structure has been proved to have a great benefit to utilize the word information and avoid the error propagation of word segmentation [7, 8]. We can

J. Lin and Q. Tang (Eds.): AC3 2022, LNICST 448, pp. 216–227, 2022.
https://doi.org/10.1007/978-3-031-17081-2_14

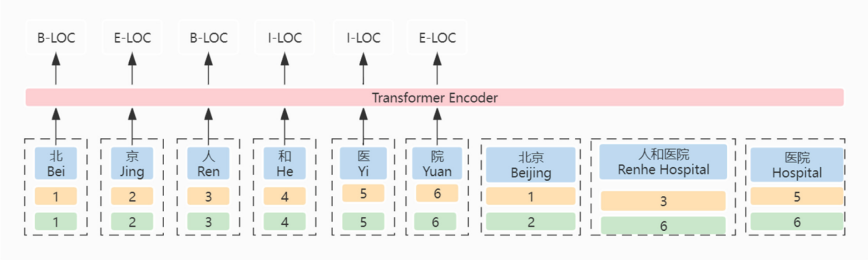
match a sentence with a lexicon to obtain the latent words in it, and then we get a lattice like in Fig. 1(a). The lattice is a directed acyclic graph, where each node is a character or a latent word. The lattice includes a sequence of characters and potential words in the sentence. They are not ordered sequentially, and the word's first character and last character determine its position. Some words in lattice may be important for



(a) Lattice.



(b) Lattice LSTM.



(c) Flat-Lattice Transformer.

Fig. 1. While lattice LSTM indicates lattice structure by dynamically adjusting its structure, FLAT only needs to leverage the span position encoding. In (c) , , , denotes tokens, heads and tails, respectively.

NER. For example, in Fig. 1(a), "人和医院(Renhe Hospital)" can be used to distinguish between the geographic entity "北京(Beijing)" and the organization entity "北京人(Beijing People)".

There are three lines of methods to leverage the lattice. (1) First line is to design a model to be compatible with lattice input, such as lattice LSTM [7] and LR-CNN [9]. In lattice LSTM, an extra word cell is employed to encode the potential words, and attention mechanism is used to fuse variable-number nodes at each position, as in Fig. 1(b). LR-CNN uses CNN to encode potential words at different window sizes. However, RNN and CNN are hard to model long-distance dependencies [10], which may be useful in NER, such as coreference [11]. Due to the dynamic lattice structure, these methods cannot fully utilize the parallel computation of GPU. (2) Second line is to convert lattice into graph and use a graph neural network (GNN) to encode it, such as Lexicon-based Graph Network (LGN) [12] and Collaborative Graph Network (CGN) [13]. While sequential structure is still important for NER and graph is general counterpart, their gap is not negligible. These methods need to use LSTM as the bottom encoder to carry the sequential inductive bias, which makes the model complicated. (3)Third line is to converts the lattice structure into a flat structure consisting of spans to encode it, such as Flat Lattice Transformer [14]. In Flat Lattice Transformer, an ingenious position encoding for the lattice-structure is designed to reconstruct a lattice from a set of tokens, as in Fig. 1(c). While word segmentation information is still important for NER, the character-word vector needs to be trained and the user-defined entity dictionary cannot be effectively used.

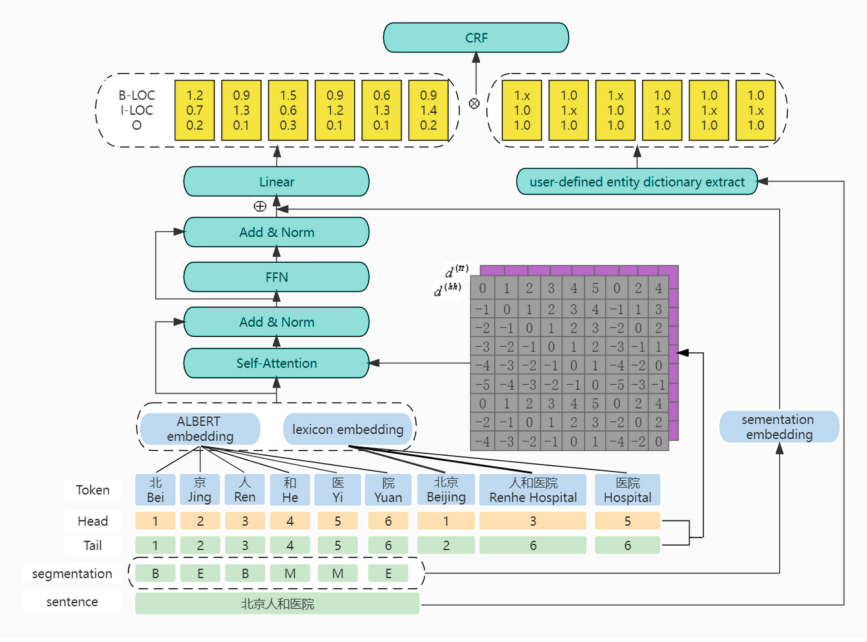


Fig. 2. The overall architecture of ALFLAT.

In this paper, we propose ALFLAT: based on a flat-lattice Transformer to incorporate ALBERT [15] pre-trained model, word segmentation information and user-defined entity dictionary for Chinese NER as shown in Fig. 2. Transformer [10] adopts fully-connected self-attention to model the long-distance dependencies in a sequence. To keep the position information, Transformer introduces the position representation for each token in the sequence. Inspired by the idea of position representation, we use an ingenious position encoding for the lattice-structure to reconstruct a lattice from a set of tokens, then we can directly use ALBERT pre-trained model and Transformer to fully model the lattice input. The self-attention mechanism of Transformer enables characters to directly interact with any potential word, including self-matched words. After obtaining the context representation of each token, we integrate word segmentation embedding with the context representation, and use a linear fully connected layer to obtain the emission matrix scores of each tag corresponding to the token, modify the emission matrix scores according to the user-defined entity dictionary, finally utilize Viterbi [16] decoding of CRF [17] to obtain the correct entities. Experimental results show our model outperforms other lexicon-based methods on the performance.

2 Background

In this section, we briefly introduce the Transformer architecture. Focusing on the NER task, we only discuss the Transformer encoder. It is composed of self-attention and feed forward network (FFN) layers. Each sublayer is followed by residual connection and layer normalization. FFN is a position-wise multi-layer Perceptron with nonlinear transformation. Transformer performs self-attention over the sequence by H heads of attention individually and then concatenates the result of H heads. For simplicity, we ignore the head index in the following formula. The result of per head is calculated as:

$$\mathrm{Att}(\mathbf{A},\mathbf{V}) = \mathrm{softmax}(\mathbf{A})\mathbf{V} \tag{1}$$

$$\mathbf{A}_{\mathrm{ij}} = \left(\frac{\mathrm{Q}_i \mathrm{K}_j^{\mathrm{T}}}{\sqrt{\mathrm{d}_{head}}}\right) \tag{2}$$

$$[\mathrm{Q},\mathrm{K},\mathrm{V}] = E_x[\mathrm{W}_q, \mathrm{W}_k, \mathrm{W}_v] \tag{3}$$

where E is the token embedding lookup table or the output of last Transformer layer. $\mathrm{W}_q, \mathrm{W}_k, \mathrm{W}_v \in \mathbb{R}^{d_{\mathrm{mod}\ el} \times d_{head}}$ are learnable parameters, and $d_{\mathrm{mod}\ el} = H \times d_{head}$, d_{head} is the dimension of each head. The vanilla Transformer also uses absolute position encoding to capture the sequential information. Inspired by [18], we think commutativity of the vector inner dot will cause the loss of directionality in self-attention. Therefore, we consider the relative position of lattice also significant for NER.

3 Model

3.1 Converting Lattice into Flat Structure

After getting a lattice from characters with a lexicon, we can flatten it into flat counterpart. The flat-lattice can be defined as a set of spans, and a span corresponds to a token, a head

and a tail, like in Fig. 1(c), proposed by [14]. The token is a character or word. The head and tail denote the position index of the token's first and last characters in the original sentence, and they indicate the position of the token in the lattice. For the character, its head and tail are the same. There is a simple algorithm to recover flat-lattice into its original structure. We can first take the token which has the same head and tail, to construct the character sequence. Then we use other tokens (words) with their heads and tails to build skip-paths. Since our transformation is recoverable, we assume flat-lattice can maintain the original structure of lattice.

3.2 Relative Position Encoding of Spans

The flat-lattice structure consists of spans with different lengths. To encode the interactions among spans, we use the relative position encoding of spans [14]. For two spans x_i and x_j in the lattice, there are three kinds of relations between them: intersection, inclusion and separation, determined by their heads and tails. Instead of directly encoding these three kinds of relations, we use a dense vector to model their relations. It is calculated by continuous transformation of the head and tail information. Thus, we think it can not only represent the relation between two tokens, but also indicate more detailed information, such as the distance between a character and a word. Let $head[i]$ and $tail[i]$ denote the head and tail position of span x_i. we only use two kinds of relative distances can be used to indicate the relation between x_i and x_j. They can be calculated as:

$$d_{ij}^{(hh)} = head[i] - head[j] \tag{4}$$

$$d_{ij}^{(tt)} = tail[i] - tail[j] \tag{5}$$

where $d_{ij}^{(hh)}$ denotes the distance between head of x_i and x_j, and $d_{ij}^{(tt)}$ denotes the distance between tail of x_i and x_j. The final relative position encoding of spans is a simple non-linear transformation of the two distances:

$$R_{ij} = \mathrm{ReLU}(W_r(\mathrm{p}_{d_{ij}^{(hh)}} \oplus \mathrm{p}_{d_{ij}^{(tt)}})) \tag{6}$$

where W_r is a learnable parameter, $\oplus$ denotes the concatenation operator, and p_d is calculated as in [10],

$$\mathrm{p}_d^{(2\kappa)} = \sin\left(d/10000^{2\kappa/d_{\mathrm{mod}\ el}}\right) \tag{7}$$

$$\mathrm{p}_d^{(2\kappa+1)} = \cos\left(d/10000^{2\kappa/d_{\mathrm{mod}\ el}}\right) \tag{8}$$

where d is $d_{ij}^{(hh)}$, $d_{ij}^{(tt)}$ and κ denotes the index of dimension of position encoding. Then we use a variant of self-attention [19] to leverage the relative span position encoding as follows:

$$\begin{aligned} \mathbf{A}_{i,j}^{*} &= \mathrm{W}_q^T \mathrm{E}_{x_i}^T \mathrm{E}_{x_\mathrm{j}} \mathrm{W}_{k,\mathrm{E}} + \mathrm{W}_q^T \mathrm{E}_{x_i}^T R_{ij} \mathrm{W}_{k,\mathrm{R}} \\ &+ \mathrm{u}^{\mathrm{T}} \mathrm{E}_{x_\mathrm{j}} \mathrm{W}_{k,\mathrm{E}} + \mathrm{v}^{\mathrm{T}} R_{ij} \mathrm{W}_{k,R} \end{aligned} \tag{9}$$

where $\mathrm{W}_q, \mathrm{W}_{\kappa,R}, \mathrm{W}_{\kappa,E} \in \mathbb{R}^{d_{\mathrm{mod}\ el} \times d_{head}}$ and u,v $\in \mathbb{R}^{d_{head}}$ are learnable parameters. Then we replace A with A* in Eq. (1). The following calculation is the same with vanilla Transformer.

3.3 Chinese Word Segmentation

Word segmentation is helpful for generating features for an NER task. Chinese word segmentation can be regarded as a character sequence labeling task, where each character in the sentence is assigned a segment label from left to right, including {B, M, E, S}, to indicate the segmentation [8]. B, M, E represent the character is the beginning, middle or end of a multi-character word, respectively. S represents that the current character is a single character word. Figure 3 gives an intuitive explaination.

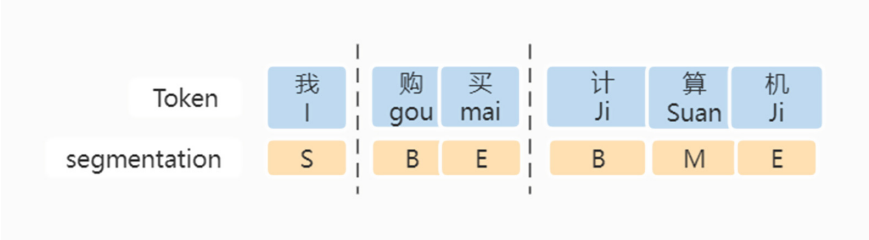


Fig. 3. Chinese word segmentation as a sequence labeling task.

In our paper, we use jieba chinese word segmentation tool [20] to assign segmentation label for each character. Given a sentence with length n: $\{w_1, w_2, ..., w_n\}$ and its corresponding segmentation labels: $\{t_1, t_2, ..., t_n\}$, each segmentation label is initialized as a d dimensional embedding. It has been shown in [21] that word embedding plays a vital role to improve sequence tagging performance. We initialize randomly the embedding which has 4 vocabulary size and each word corresponds to a d-dimensional embedding vector. To use this embedding, the segmentation labels of the given sentence were transform into embeddings. This transformation is done by lookup table operation. This embeddings were modified during its training. After the lookup table operation, we obtained a matrix $X \in \mathbb{R}^{n \times d}$, where the *i'th* row is the segmentation label embedding of t_i.

Next, we integrate segmentation embeddings with the output of flat-lattice transformer module, which is defined as:

$$H' = H \oplus X \tag{10}$$

where H denotes the output of the flat-lattice transformer module, $\oplus$ denotes the concatenation operation.

Then the output of concatenation operation was fed to a fully connected linear model, which projects each token's encoded representation to the tag space.

3.4 Modify the Emission Matrix Scores

This module aims to extract all named entities from the input original sentence accord to the user-defined entity dictionary, including the entity name and the corresponding entity type. For example, given a sentence with n length:$\{w_1, w_2, ..., w_n\}$,we extract the entity

labels:$\{a_1, a_2, ..., a_n\}$ according to the user-defined entity dictionary. Then we initialize a matrix $B \in \mathbb{R}^{n\times K}$ with element 1, where K is the number of name entities tags, and appropriately modify the matrix B, let $B_{ij} = 1.x$, where $i \in \{1, ..., n\}$, j denotes the index of label a_i, which a_i is not label 'O'. The modification of emission matrix scores process can be define as:

$$P^{'} = P \otimes B \tag{11}$$

where $\otimes$ denotes the multiplication operation, P denotes the output of the fully connected linear model, named as emission matrix scores $p \in \mathbb{R}^{n\times K}$. This module serves as an auxiliary method to improve the effect of entity recognition, and is only used in evaluation process.

3.5 Conditional Random Fields

If the emission matrix scores learned by ALFLAT will deviate from the actual, then we need to introduce CRF to optimize it. It happens that CRF can constrain between tags and tags. There is a dependency between them, and the use of CRF can reduce errors. For an input sequence of n tokens, we integrate segmentation embeddings with the output of flat-lattice transformer model, outputs an encoded token sequence with hidden dimension $d_{\text{mod } el}$. The fully connected layer linear model projects each token's encoded representation to the tag space, i.e. $\mathbb{R}^{d_{\text{mod } el}} \mapsto \mathbb{R}^K$, where K is the number of name entity tags. The emission matrix scores $p \in \mathbb{R}^{n\times K}$ of the linear model are then modified according to the user-defined entity dictionary. Finally, the new emission matrix scores $p^{'} \in \mathbb{R}^{n\times K}$ fed to the CRF layer, whose parameters are a matrix of tag transitions $A \in \mathbb{R}^{K\times K}$. The matrix A is such that $A_{i,j}$ represents the score of transitioning from tag i to tag j. As described by [22], for an input sequence $X = (x_1,...,x_n)$ and a sequence of tag predictions $y = (y_1,...,y_n)$, $y_i \in \{1, ..., K\}$, the score of the sequence is defined as

$$s(X, y) = \sum_{i=0}^{n} A_{y_i, y_{i+1}} + \sum_{i=1}^{n} p_{i,y_i} \tag{12}$$

where y_0 and y_{n+1} are start and end tags. A softmax over all possible tag sequences yields a probability for the sequence y:

$$p(y|X) = \frac{e^{s(X,y)}}{\sum\limits_{y^{'}\in Y_X} e^{s(X,y^{'})}} \tag{13}$$

During training, we maximize the log-probability of the correct tag sequence:

$$\log(p(y|X)) = s(X, y) - \log\left(\sum_{y^{'}\in Y_X} e^{s(X,y^{'})}\right) \tag{14}$$

where Y_X are all possible tag sequences. The summation in Eq. 14 is computed using dynamic programming. During evaluation, we predict the output sequence that obtains

the maximum score given by:

$$y^* = \arg\max_{y' \in Y_X} s(X, y') \tag{15}$$

This can be computed using the Viterbi decoding [16].

4 Experiments

4.1 Experimental Setup

In order to evaluate our model, we used **MSRA** [23] Chinese NER dataset and show statistics of the dataset in Table 1. We take TENER [18], BiLSTM-CRF as baseline models, and use the same train, test, dev split as Gui [12]. The embeddings and lexicons are the same as Zhang and Yang [7]. We select the hyper-parameters are the same as Li [14], which based on the development experiment of MSRA. The Table 2 lists the hyper-parameters obtained from the development experiment of MSRA. In particular, we use only one layer Transformer encoder for our model.

Table 1. Statistics of MSRA dataset. 'Train' is the size of training set. 'Char_{avg}', 'Word_{avg}', '$\text{Entity}_{\text{avg}}$' are the average number of chars, words matched by lexicon and entities in an instance.

Name	Size
Train	46675
Char_{avg}	45.87
Word_{avg}	22.38
$\text{Entity}_{\text{avg}}$	1.58

In terms of evaluation metrics, we not only use the common F1 score, but also use two metrics proposed by Li [14], including **Span F** and **Type Acc.**

4.2 Overall Performance

As shown in Table 3, our model outperforms baseline models and other lexicon-based models on MSRA Chinese NER dataset. Our model outperforms TENER [18] by 3.54 in average F1 score. For lattice LSTM, our model has an average F1 improvement of 3.27 over it. For FLAT [14], our model has an average F1 improvement of 2.33 over it. We also compare ALFLAT with BERT + FLAT tagger on MSAR dataset, our model also outperforms it by 0.34 in average F1 score.

Table 2. Hyper-parameters for ALFLAT model.

Name	Value
lr	1e-3
-decay	0.05
optimizer	SGD
-momentum	0.9
d_{model}	160
head	8
FFN size	480
Embed dropout	0.5
Output dropout	0.3
warmup	10(epoch)

Table 3. MSRA dataset results (F1). BiLSTM results are from Zhang and Yang [7]. 'YJ' denotes the lexicon released by Zhang and Yang [7]. 'FLAT + BERT' results are from Li [14]. The result of other models are from their original paper. ALFLAT uses ALBERT embedding. We finetune ALBERT in our model during training. The ALBERT in the experiment is 'albert_base_zh' released by albert [25]. We use it by the AlbertModel embedding in transformers [26].

Model	Lexicon	F1 score
BiLSTM	-	91.84
TENER	-	92.89
Lattice LSTM	YJ	93.16
FLAT	YJ	94.10
FLAT + BERT	YJ	96.09
ALFLAT	YJ	**96.43**

4.3 How ALFLAT Use ALBERT Instead of BERT

ALBERT architecture has significantly fewer parameters than a traditional BERT architecture. ALBERT incorporates two parameter reduction techniques that lift the major obstacles in scaling pre-trained models. Both two techniques significantly reduce the number of parameters for BERT without seriously hurting performance, thus improving parameter-efficiency. An ALBERT configuration similar to BERT-large has 18x fewer parameters and can be trained about 1.7x faster. The parameter reduction techniques also act as a form of regularization that stabilizes the training and helps with generalization.

4.4 How ALFLAT Brings Improvement

Table 4 shows the **Span F** and the **Type Acc** of four models on the evaluation set of MSRA. We can find: 1) ALFLAT outperforms FLAT in two metrics significantly. 2) The improvement on **Span F** brought by ALFLAT is more significant than that on **Type Acc**. These show: 1) The position encoding helps ALFLAT locate entities more accurately. 2) The pre-trained word embedding makes ALFLAT more powerful in entity recognition [24].

Table 4. Two metrics of models. FLAT results are from Li [14]. The result of other models are from their original paper. Except that the superscript * means the result is not provided in the original paper, and we get the result by running the public source code.

Model	Span F	Type Acc
TENER	93.17	99.29
FLAT	94.58	99.52
FLAT + BERT	96.26*	99.38*
ALFLAT	**96.64**	**99.56**

5 Conclusion

In this paper, we introduce a new Chinese NER method according to use a flat-lattice Transformer to incorporate ALBERT pre-trained model, user-defined entity dictionary, word segmentation information. The core of our model is using a set of spans of lattice structure, the specific position encoding, word segmentation and modifying the emission matrix scores according to user-defined entity dictionary. Experimental results show our model outperforms other lexicon based models in the performance. We leave adjusting our model to different kinds of lattice or graph as our future work.

Funding. The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This article is supported in part by the National Key R&D Program of China under project (2020YFB1006003), the National Natural Science Foundation of China under project (62172119), the Guangxi Natural Science Foundation under grant (2019GXNSFGA245004), and the Major Key Project of PCL under grants (PCL2022A03, PCL2021A02, PCL2021A09).

References

1. Chen, Y., Xu, L., Liu, K., Zeng, D., Zhao, J.: Event extraction via dynamic multipooling 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, Beijing, China, pp. 167–176.Association for Computational Linguistics (2015)

2. Diefenbach, D., Lopez, V., Singh, K., Maret, P.: Core techniques of question answering systems over knowledge bases: a survey. Knowl. Inf. Syst. **55**(3), 529–569 (2017). https://doi.org/10.1007/s10115-017-1100-y
3. Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C.: Neural architectures for named entity recognition. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego, California, pp. 260–270. Association for Computational Linguistics (2016)
4. Yang, J., Zhang, Y., Dong, F.: Neural reranking for named entity recognition. arXiv preprint arXiv:1707.05127 (2017)
5. Liu, L., et al.: Empower sequence labeling with task-aware neural language model. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, no. 1 (2018)
6. Sun, T., Shao, Y., Li, X., Liu, P., Yan, H., Qiu, X., Huang, X.: Learning sparse sharing architectures for multiple tasks. In: Proceedings of the AAAI Conference on Artificial Intelligence,vol. 34, no. 5, pp. 8936–8943 (2020)
7. Zhang, Y., Yang, J.: Chinese NER using lattice LSTM. arXiv preprint arXiv:1805.02023 (2018)
8. Zhao, H., Huang, C.-N., Li, M.: An improved Chinese word segmentation system with conditional random field. In: Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing, Sydney, pp. 162–165 (2006)
9. Gui, T., Ma, R., Zhang, Q., Zhao, L., Jiang, Y.-G., Huang, X.: CNN-based Chinese NER with lexicon rethinking. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI 2019, pp. 4982–4988. AAAI Press (2019)
10. Vaswani, A., et al.: Attention is all you need. In: Guyon, I., et al. (eds.) Advances in Neural Information Processing Systems, vol. 30, pp. 5998–6008. Curran Associates, Inc. (2017)
11. Stanislawek, T., Wróblewska, A., Wójcicka, A., Ziembicki, D., Biecek, P.: Named entity recognition - is there a glass ceiling? In: Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL), Hong Kong, China, pp. 624–633. Association for Computational Linguistics (2019)
12. Gui, T., Zou, Y., Zhang, Q., Peng, M., Fu, J., Wei, Z., Huang, X.: 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), Hong Kong, China, pp. 1039–1049. Association for Computational Linguistics (2019)
13. Sui, D., Chen, Y., Liu, K., Zhao, J., Liu, S.: 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 (EMNLPIJCNLP), Hong Kong, China, pp. 3821–3831. Association for Computational Linguistics (2019)
14. Li, X., et al. FLAT: Chinese NER using flat-lattice transformer. arXiv preprint arXiv:2004.11795 (2020)
15. Lan, Z., et al. ALBERT: a lite BERT for self-supervised learning of language representations. arXiv preprint arXiv:1909.11942 (2019)
16. Viterbi, A.J., et al.: A pragmatic approach to trellis-coded modulation. IEEE Commun. Mag. **27**(7), 11–19 (1989)
17. Lafferty, J., McCallum, A., Pereira, F.C.N.: Conditional random fields: probabilistic models for segmenting and labeling sequence data (2001)
18. Yan, H., et al.: TENER: adapting transformer encoder for named entity recognition. arXiv preprint arXiv:1911.04474 (2019)
19. Dai, Z., et al.: Transformer-xl: attentive language models beyond a fixed-length context. arXiv preprint arXiv:1901.02860 (2019)

20. Sun, J.: Jieba Chinese word segmentation tool (2012)
21. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. J. Mach. Learn. Res. **12**(1), 2493–2537 (2011)
22. Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C.: Neural architectures for named entity recognition. Computing Research Repository arXiv:1603.01360 (2016)
23. Levow, G.-A.: The third international Chinese language processing bakeoff: Word segmentation and named entity recognition. In: Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing, Sydney, Australia, pp. 108–117. Association for Computational Linguistics (2006)
24. Agarwal, O., Yang, Y., Wallace, B., Nenkova, A.: Interpretability analysis for named entity recognition to understand system predictions and how they can improve. Comput. Linguist. **47**(1), 117–140 (2021)
25. ALBERT Homepage. https://github.com/google-research/albert
26. TRANSFORMERS Homepage. https://github.com/huggingface/transformers