FLAT: Chinese NER Using Flat-Lattice Transformer

Xiaonan Li, Hang Yan, Xipeng Qiu[*], Xuanjing Huang
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
School of Computer Science, Fudan University
lixiaonan_xdu@outlook.com, {hyan19, xpqiu, xjhuang}@fudan.edu.cn

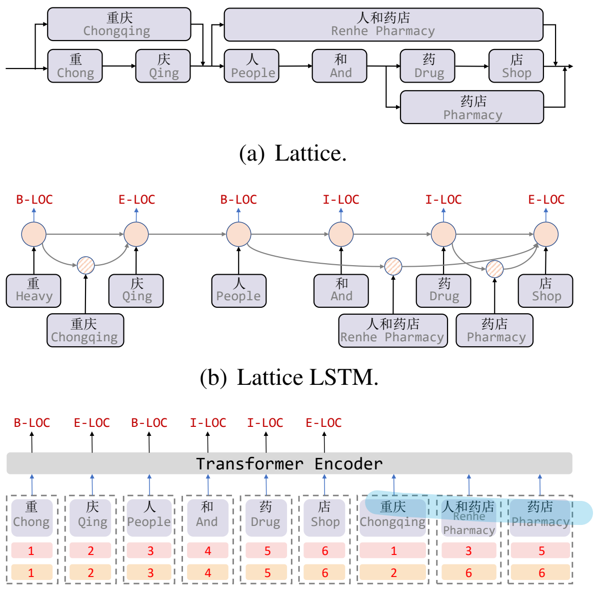
Abstract

Recently, the character-word lattice structure has been proved to be effective for Chinese named entity recognition (NER) by incorporating the word information. However, since the lattice structure is complex and dynamic, most existing lattice-based models are hard to fully utilize the parallel computation of GPUs and usually have a low inference-speed. In this paper, we propose **FLAT**: **F**lat-**LA**ttice **T**ransformer for Chinese NER, which converts the lattice structure into a flat structure consisting of spans. Each span corresponds to a character or latent word and its position in the original lattice. With the power of Transformer and well-designed position encoding, FLAT can fully leverage the lattice information and has an excellent parallelization ability. Experiments on four datasets show FLAT outperforms other lexicon-based models in performance and efficiency.

1 Introduction

Named entity recognition (NER) plays an indispensable role in many downstream natural language processing (NLP) tasks (Chen et al., 2015; Diefenbach et al., 2018). Compared with English NER (Lample et al., 2016; Yang et al., 2017; Liu et al., 2017; Sun et al., 2020), 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 (Zhang and Yang, 2018). We can match a sentence with a lexicon to obtain the latent words in it, and then we get a lattice like in Figure 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 NER. For example, in Figure 1(a), "人和药店(Renhe Pharmacy)" can be used to distinguish between the geographic entity "重庆(Chongqing)" and the organization entity "重庆人(Chongqing People)".



(a) Lattice.

(b) Lattice LSTM.

(c) Flat-Lattice Transformer.

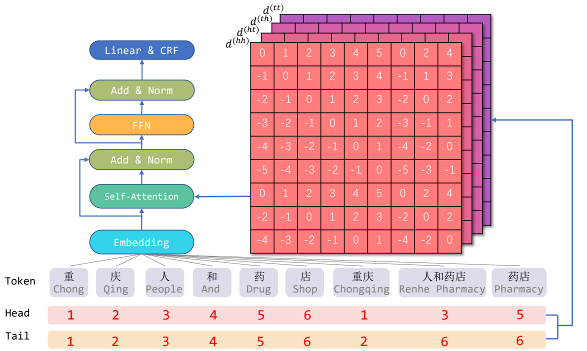
Figure 1: While lattice LSTM indicates lattice structure by dynamically adjusting its structure, FLAT only needs to leverage the span position encoding. In 1(c), 🟪, 🟥, 🟧 denotes tokens, heads and tails, respectively.

There are two lines of methods to leverage the lattice. (1) One line is to design a model to be compatible with lattice input, such as lattice LSTM (Zhang and Yang, 2018) and LR-CNN (Gui et al., 2019a). 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 Figure 1(b). LR-CNN uses

[*] Corresponding author.

CNN to encode potential words at different window sizes. However, RNN and CNN are hard to model long-distance dependencies (Vaswani et al., 2017), which may be useful in NER, such as coreference (Stanislawek et al., 2019). Due to the dynamic lattice structure, these methods cannot fully utilize the parallel computation of GPU. (2) Another line is to convert lattice into graph and use a graph neural network (GNN) to encode it, such as Lexicon-based Graph Network (LGN) (Gui et al., 2019b) and Collaborative Graph Network (CGN) (Sui et al., 2019). 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.

In this paper, we propose **FLAT**: Flat **LA**ttice **T**ransformer for Chinese NER. Transformer (Vaswani et al., 2017) 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 design an ingenious position encoding for the lattice-structure, as shown in Figure 1(c). In detail, we assign two positional indices for a token (character or word): head position and tail position, by which we can reconstruct a lattice from a set of tokens. Thus, we can directly use 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. To a character, its self-matched words denote words which include it. For example, in Figure 1(a), self-matched words of "药 (Drug)" are "人和药店(Renhe Pharmacy)" and "药店 (Pharmacy)"(Sui et al., 2019). Experimental results show our model outperforms other lexicon-based methods on the performance and inference-speed.



0	1	2	3	4	5	0	2	4
-1	0	1	2	3	4	-1	1	3
-2	-1	0	1	2	3	-2	0	2
-3	-2	-1	0	1	2	-3	-1	1
-4	-3	-2	-1	0	1	-4	-2	0
-5	-4	-3	-2	-1	0	-5	-3	-1
0	1	2	3	4	5	0	2	4
-2	-1	0	1	2	3	-2	0	2
-4	-3	-2	-1	0	1	-4	-2	0

Token	重 Chong	庆 Qing	人 People	和 And	药 Drug	店 Shop	重庆 Chongqing	人和药店 Renhe Pharmacy	药店 Pharmacy
Head	1	2	3	4	5	6	1	3	5
Tail	1	2	3	4	5	6	2	6	6

Figure 2: The overall architecture of FLAT.

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 feedforward network (FFN) layers. Each sublayer is followed by residual connection and layer normalization. FFN is a position-wise multi-layer Perceptron with non-linear 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:

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

$$\mathbf{A_{ij}} = \left(\frac{\mathbf{Q_i K_j}^{\mathrm{T}}}{\sqrt{\mathrm{d_{head}}}}\right), \tag{2}$$

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

where E is the token embedding lookup table or the output of last Transformer layer. $\mathbf{W}_{\mathrm{q}}, \mathbf{W}_{\mathrm{k}}, \mathbf{W}_{\mathrm{v}} \in \mathbb{R}^{d_{model} \times d_{head}}$ are learnable parameters, and $d_{model} = H \times d_{head}$, d_{head} is the dimension of each head.

The vanilla Transformer also uses absolute position encoding to capture the sequential information. Inspired by Yan et al. (2019), 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 Figure 1(c). 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 sequence, 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 propose the relative position encoding of spans. 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. Four kinds of relative distances can be used to indicate the relation between x_i and x_j. They can be calculated as:

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

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

$$d_{ij}^{(th)} = tail[i] - head[j], \tag{6}$$

$$d_{ij}^{(tt)} = tail[i] - tail[j], \tag{7}$$

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

$$R_{ij} = \mathrm{ReLU}(W_r(\mathbf{p}_{d_{ij}^{(hh)}} \oplus \mathbf{p}_{d_{ij}^{(th)}} \oplus \mathbf{p}_{d_{ij}^{(ht)}} \oplus \mathbf{p}_{d_{ij}^{(tt)}})), \tag{8}$$

where W_r is a learnable parameter, $\oplus$ denotes the concatenation operator, and $\mathbf{p}_d$ is calculated as in Vaswani et al. (2017),

$$\mathbf{p}_d^{(2k)} = \sin\left(d/10000^{2k/d_{model}}\right), \tag{9}$$

$$\mathbf{p}_d^{(2k+1)} = \cos\left(d/10000^{2k/d_{model}}\right), \tag{10}$$

where d is $d_{ij}^{(hh)}$, $d_{ij}^{(ht)}$, $d_{ij}^{(th)}$ or $d_{ij}^{(tt)}$ and k denotes the index of dimension of position encoding. Then we use a variant of self-attention (Dai et al., 2019) to leverage the relative span position encoding as follows:

$$\begin{aligned}\mathbf{A}_{i,j}^{*} &= \mathbf{W}_q^\top \mathbf{E}_{x_i}^\top \mathbf{E}_{x_j} \mathbf{W}_{k,E} + \mathbf{W}_q^\top \mathbf{E}_{x_i}^\top \mathbf{R}_{ij} \mathbf{W}_{k,R} \\ &+ \mathbf{u}^\top \mathbf{E}_{x_j} \mathbf{W}_{k,E} + \mathbf{v}^\top \mathbf{R}_{ij} \mathbf{W}_{k,R},\end{aligned} \tag{11}$$

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

	Ontonotes	MSRA	Resume	Weibo
Train	15740	46675	3821	1350
Char$_{avg}$	36.92	45.87	32.15	54.37
Word$_{avg}$	17.59	22.38	24.99	21.49
Entity$_{avg}$	1.15	1.58	3.48	1.42

Table 1: Statistics of four datasets. ‘Train’ is the size of training set. ‘Char$_{avg}$’, ‘Word$_{avg}$’, ‘Entity$_{avg}$’ are the average number of chars, words mateched by lexicon and entities in an instance.

	Lexicon	Ontonotes	MSRA	Resume	Weibo
BiLSTM	-	71.81	91.87	94.41	56.75
TENER	-	72.82	93.01	95.25	58.39
Lattice LSTM	YJ	73.88	93.18	94.46	58.79
CNNR	YJ	74.45	93.71	95.11	59.92
LGN	YJ	74.85	93.63	95.41	60.15
PLT	YJ	74.60	93.26	95.40	59.92
FLAT	YJ	**76.45**	**94.12**	**95.45**	**60.32**
FLAT$_{\mathrm{msm}}$	YJ	73.39	93.11	95.03	57.98
FLAT$_{\mathrm{mld}}$	YJ	75.35	93.83	95.28	59.63
CGN	LS	74.79	93.47	94.12*	63.09
FLAT	LS	**75.70**	**94.35**	**94.93**	**63.42**

Table 2: Four datasets results (F1). BiLSTM results are from Zhang and Yang (2018). PLT denotes the porous lattice Transformer (Mengge et al., 2019). ‘YJ’ denotes the lexicon released by Zhang and Yang (2018), and ‘LS’ denotes the lexicon released by Li et al. (2018). 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. Subscripts ‘msm’ and ‘mld’ denote FLAT with the mask of self-matched words and long distance (>10), respectively.

After FLAT, we only take the character representation into output layer, followed by a Condiftional Random Field (CRF) (Lafferty et al., 2001).

	Span F		Type Acc	
	Ontonotes	MSRA	Ontonotes	MSRA
TENER	72.41	93.17	96.33	99.29
FLAT	76.23	94.58	97.03	99.52
FLAT$_{head}$	75.64	94.33	96.85	99.45

Table 3: Two metrics of models. FLAT$_{head}$ means R_{ij} in (11) is replaced by $d_{ij}^{(hh)}$.

4 Experiments

4.1 Experimental Setup

Four Chinese NER datasets were used to evaluate our model, including (1) **Ontonotes 4.0** (Weischedel and Consortium, 2013) (2) **MSRA** (Levow, 2006) (3) **Resume** (Zhang and Yang, 2018) (4) **Weibo** (Peng and Dredze, 2015; He and Sun, 2016). We show statistics of these datasets in Table 1. We use the same train, dev, test split as Gui et al. (2019b). We take BiLSTM-CRF and TENER (Yan et al., 2019) as baseline models. TENER is a Transformer using relative position encoding for NER, without external information. We also compare FLAT with other lexicon-based methods. The embeddings and lexicons are the same as Zhang and Yang (2018). When comparing with CGN (Li et al., 2018), we use the same lexicon as CGN. The way to select hyper-parameters can be found in the supplementary material. In particular, we use only one layer Transformer encoder for our model.

4.2 Overall Performance

As shown in Table 2, our model outperforms baseline models and other lexicon-based models on four Chinese NER datasets. Our model outperforms TENER (Yan et al., 2019) by 1.72 in average F1 score. For lattice LSTM, our model has an average F1 improvement of 1.51 over it. When using another lexicon (Li et al., 2018), our model also outperforms CGN by 0.73 in average F1 score. Maybe due to the characteristic of Transformer, the improvement of FLAT over other lexicon-based models on small datasets is not so significant like that on large datasets.

4.3 Advantage of Fully-Connected Structure

We think self-attention mechanism brings two advantages over lattice LSTM: 1) All characters can directly interact with its self-matched words. 2) Long-distance dependencies can be fully modeled. Due to our model has only one layer, we can strip them by masking corresponding attention. In detail, we mask attention from the character to its self-matched word and attention between tokens whose distance exceeds 10. As shown in Table 2, the first mask brings a significant deterioration to FLAT while the second degrades performance slightly. As a result, we think leveraging information of self-matched words is important For Chinese NER.

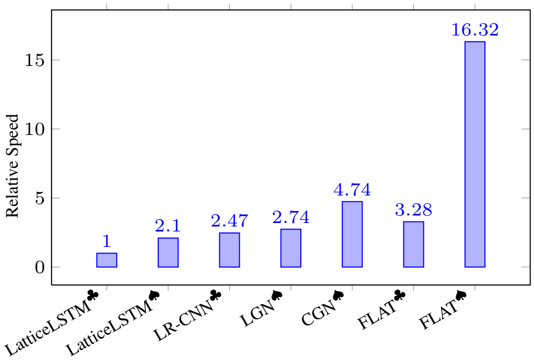


Figure 3: Inference-speed of different models, compared with lattice LSTM ♣. ♣ denotes non-batch-parallel version, and ♠ indicates the model is run in 16 batch size parallelly. For model LR-CNN, we do not get its batch-parallel version.

4.4 Efficiency of FLAT

To verify the computation efficiency of our model, we compare the inference-speed of different lexicon-based models on Ontonotes. The result is shown in Figure 3. GNN-based models outperform lattice LSTM and LR-CNN. But the RNN encoder of GNN-based models also degrades their speed. Because our model has no recurrent module and can fully leverage parallel computation of GPU, it outperforms other methods in running efficiency. In terms of leveraging batch-parallelism, the speedup ratio brought by batch-parallelism is 4.97 for FLAT, 2.1 for lattice LSTM, when batch size = 16. Due to the simplicity of our model, it can benefit from batch-parallelism more significantly.

4.5 How FLAT Brings Improvement

Compared with TENER, FLAT leverages lexicon resources and uses a new position encoding. To probe how these two factors bring improvement. We set two new metrics, 1) **Span F**: while the common F score used in NER considers correctness of both the span and the entity type, Span F only considers the former. 2) **Type Acc**: proportion of full-correct predictions to span-correct predictions. Table 3 shows two metrics of three models on the devlopment set of Ontonotes and MSRA. We can find: 1) FLAT outperforms TENER in two metrics significantly. 2) The improvement on Span F brought by FLAT is more significant than that on Type Acc. 3) Compared to FLAT, FLAT$_{head}$'s deterioration on Span F is more significant than that on Type Acc. These show: 1) The new position encoding helps FLAT locate entities more accurately. 2) The pre-trained word-level embedding

	Lexicon	Ontonotes	MSRA	Resume	Weibo
BERT	-	80.14	94.95	95.53	68.20
BERT+FLAT	YJ	81.82	96.09	95.86	68.55

Table 4: Comparision between BERT and BERT+FLAT. 'BERT' refers to the BERT+MLP+CRF architecture. 'FLAT+BERT' refers to FLAT using BERT embedding. We finetune BERT in both models during training. The BERT in the experiment is 'BERT-wwm' released by Cui et al. (2019). We use it by the BERTEmbedding in fastNLP [1].

makes FLAT more powerful in entity classification (Agarwal et al., 2020).

4.6 Compatibility with BERT

We also compare FLAT equipped with BERT with common BERT+CRF tagger on four datasets, and Results are shown in Table 4. We find that, for large datasets like Ontonotes and MSRA, FLAT+BERT can have a significant improvement over BERT. But for small datasets like Resume and Weibo, the improvement of FLAT+BERT over BERT is marginal.

5 Related Work

5.1 Lexicon-based NER

Zhang and Yang (2018) introduced a lattice LSTM to encode all characters and potential words recognized by a lexicon in a sentence, avoiding the error propagation of segmentation while leveraging the word information. Gui et al. (2019a) exploited a combination of CNN and rethinking mechanism to encode character sequence and potential words at different window sizes. Both models above suffer from the low inference efficiency and are hard to model long-distance dependencies. Gui et al. (2019b) and Sui et al. (2019) leveraged a lexicon and character sequence to construct graph, converting NER into a node classification task. However, due to NER's strong alignment of label and input, their model needs an RNN module for encoding. The main difference between our model and models above is that they modify the model structure according to the lattice, while we use a well-designed position encoding to indicate the lattice structure.

5.2 Lattice-based Transformer

For lattice-based Transformer, it has been used in speech translation and Chinese-source translation. The main difference between them is the way to indicate lattice structure. In Chinese-source translation, Xiao et al. (2019) take the absolute position of nodes' first characters and the relation between each pair of nodes as the structure information. In speech translation, Sperber et al. (2019) used the longest distance to the start node to indicate lattice structure, and Zhang et al. (2019) used the shortest distance between two nodes. Our span position encoding is more natural, and can be mapped to all the three ways, but not vise versa. Because NER is more sensitive to position information than translation, our model is more suitable for NER. Recently, Porous Lattice Transformer (Mengge et al., 2019) is proposed for Chinese NER. The main difference between FLAT and Porus Lattice Transformer is the way of representing position information. We use 'head' and 'tail' to represent the token's position in the lattice. They use 'head', tokens' relative relation (not distance) and an extra GRU. They also use 'porous' technique to limit the attention distribution. In their model, the position information is not recoverable because 'head' and relative relation can cause position information loss. Briefly, relative distance carries more information than relative relation.

6 Conclusion and Future Work

In this paper, we introduce a flat-lattice Transformer to incorporate lexicon information for Chinese NER. The core of our model is converting lattice structure into a set of spans and introducing the specific position encoding. Experimental results show our model outperforms other lexicon-based models in the performance and efficiency. We leave adjusting our model to different kinds of lattice or graph as our future work.

Acknowledgments

We thank anonymous reviewers for their responsible attitude and helpful comments. We thank Tianxiang Sun, Yunfan Shao and Lei Li for their help, such as drawing skill sharing, pre-reviewing, etc. This work is supported by the National Key Research and Development Program of China (No. 2018YFC0831103), National Natural Science Foundation of China (No. U1936214 and 61672162), Shanghai Municipal Science and Technology Major Project (No. 2018SHZDZX01) and ZJLab.

[1] https://github.com/fastnlp/fastNLP

References

Oshin Agarwal, Yinfei Yang, Byron Wallace, and Ani Nenkova. 2020. Interpretability analysis for named entity recognition to understand system predictions and how they can improve.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 167–176, Beijing, China. Association for Computational Linguistics.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Ziqing Yang, Shijin Wang, and Guoping Hu. 2019. Pre-training with whole word masking for chinese BERT. *CoRR*, abs/1906.08101.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *CoRR*, abs/1901.02860.

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

Tao Gui, Ruotian Ma, Qi Zhang, Lujun Zhao, Yu-Gang Jiang, and Xuanjing Huang. 2019a. Cnn-based chinese ner with lexicon rethinking. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, IJCAI'19, pages 4982–4988. AAAI Press.

Tao Gui, Yicheng Zou, Qi Zhang, Minlong Peng, Jinlan Fu, Zhongyu Wei, and Xuanjing Huang. 2019b. A lexicon-based graph neural network for Chinese NER. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1039–1049, Hong Kong, China. Association for Computational Linguistics.

Hangfeng He and Xu Sun. 2016. F-score driven max margin neural network for named entity recognition in chinese social media. *CoRR*, abs/1611.04234.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. 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*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Gina-Anne Levow. 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*, pages 108–117, Sydney, Australia. Association for Computational Linguistics.

Shen Li, Zhe Zhao, Renfen Hu, Wensi Li, Tao Liu, and Xiaoyong Du. 2018. Analogical reasoning on Chinese morphological and semantic relations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 138–143, Melbourne, Australia. Association for Computational Linguistics.

Liyuan Liu, Jingbo Shang, Frank F. Xu, Xiang Ren, Huan Gui, Jian Peng, and Jiawei Han. 2017. Empower sequence labeling with task-aware neural language model. *CoRR*, abs/1709.04109.

Xue Mengge, Yu Bowen, Liu Tingwen, Wang Bin, Meng Erli, and Li Quangang. 2019. Porous lattice-based transformer encoder for chinese ner.

Nanyun Peng and Mark Dredze. 2015. Named entity recognition for Chinese social media with jointly trained embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 548–554, Lisbon, Portugal. Association for Computational Linguistics.

Matthias Sperber, Graham Neubig, Ngoc-Quan Pham, and Alex Waibel. 2019. Self-attentional models for lattice inputs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1185–1197, Florence, Italy. Association for Computational Linguistics.

Tomasz Stanislawek, Anna Wróblewska, Alicja Wójcicka, Daniel Ziembicki, and Przemyslaw Biecek. 2019. Named entity recognition - is there a glass ceiling? In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 624–633, Hong Kong, China. Association for Computational Linguistics.

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

Tianxiang Sun, Yunfan Shao, Xiaonan Li, Pengfei Liu, Hang Yan, Xipeng Qiu, and Xuanjing Huang. 2020. Learning sparse sharing architectures for multiple tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Ralph M Weischedel and Linguistic Data Consortium. 2013. Ontonotes release 5.0. Title from disc label.

Fengshun Xiao, Jiangtong Li, Hai Zhao, Rui Wang, and Kehai Chen. 2019. Lattice-based transformer encoder for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3090–3097, Florence, Italy. Association for Computational Linguistics.

Hang Yan, Bocao Deng, Xiaonan Li, and Xipeng Qiu. 2019. Tener: Adapting transformer encoder for named entity recognition.

Jie Yang, Yue Zhang, and Fei Dong. 2017. Neural reranking for named entity recognition. *CoRR*, abs/1707.05127.

Pei Zhang, Niyu Ge, Boxing Chen, and Kai Fan. 2019. Lattice transformer for speech translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6475–6484, Florence, Italy. Association for Computational Linguistics.

Yue Zhang and Jie Yang. 2018. Chinese NER using lattice LSTM. *CoRR*, abs/1805.02023.

7 Appendices

7.1 Hyperparameters Selection

For MSRA and Ontonotes these two large datasets, we select the hyper-parameters based on the development experiment of Ontonotes. For two small datasets, Resume and Weibo, we find their optimal hyper-parameters by random-search. The Table 5 lists the hyper-parameters obtained from the development experiment of Ontonotes.

The Table 6 lists the range of hyper-parameters random-search for Weibo, Resume datasets. For the hyper-parameters which do not appear in it, they are the same as in Table 5.

batch	10
lr	1e-3
-decay	0.05
optimizer	SGD
-momentum	0.9
d$_{m}odel$	160
head	8
FFN size	480
embed dropout	0.5
output dropout	0.3
warmup	10 (epoch)

Table 5: Hyper-parameters for Ontonotes and MSRA.

batch	[8,10]
lr	[1e-3, 8e-4]
d$_{h}ead$	[16,20]
head	[4,8,12]
warmup	[1, 5, 10] (epoch)

Table 6: The range of hyper-parameters random-search for Weibo, Resume datasets.