Porous Lattice Transformer Encoder for Chinese NER

Anonymous ACL submission

Abstract

Incorporating lexicons into character-level Chinese NER by lattices is proven effective to exploit rich word boundary information. Previous work has extended RNNs to consume lattice inputs and achieved great success. However, due to the DAG structure and the inherently unidirectional sequential nature, this method precludes batched computation and sufficient semantic interaction. In this paper, we propose PLTE, an extension of transformer encoder that is tailored for Chinese NER, which models all the characters and matched lexical words in parallel with batch processing. PLTE augments self-attention with positional relation representations to incorporate lattice structure. It also introduces a porous mechanism to augment localness modeling and maintain the strength of capturing the rich long-term dependencies. Experimental results show that PLTE performs up to 11.4 times faster than state-of-the-art methods while realizing better performance. We also demonstrate that using BERT representations further substantially boosts the performance and brings out the best in PLTE. [1]

1 Introduction

Named Entity Recognition (NER) is one of the core tasks in natural language processing (NLP), which aims to automatically discover information entities and identify their corresponding categories from plain text. Many high-level NLP tasks, such as information retrieval (Berger and Lafferty, 2017), relation extraction (Yu et al., 2019) and entity linking (Xue et al., 2019), need the NER tagger as one of their essential components.

Recent studies show that English NER models have achieved great performance by integrating character information into word representations based on sequence labeling. However, different from English NER, many East Asian languages, including Chinese, are written without explicit word boundary, resulting a much tricky NER task. One intuitive way is to segment the input sentences into words first, and then to apply word sequence labeling (Yang et al., 2016; He and Sun, 2017a). Such framework makes the task easy to conduct, but ignores the underlying interactions and error propagation between these two subtasks.

[1]The source code of this paper can be obtained from https://github.com/xxx/xxx.

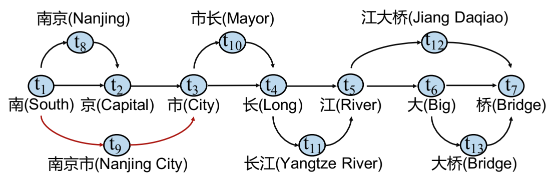


Figure 1: Example of word character lattice. Restricted by the unidirectional sequential nature, lattice LSTM has no ability to model the semantic interaction between word "南京市(Nanjing City)" and its constituent characters "南(South)" and "京(Capital)", resulting in the loss of crucial information for tagging. In addition, lattice LSTM cannot perform batched computation due to the directed acyclic graph input structure.

To overcome this limitation, many efforts have been devoted to incorporating word information by leveraging lexicon features (Peng and Dredze, 2015; Cao et al., 2018; Wu et al., 2019), especially the recent state-of-the-art (SOTA) lattice-based methods (Zhang and Yang, 2018). In particular, Zhang and Yang (2018) integrated matched lexical words information into character sequence with a directed acyclic graph (DAG) structure using lattice LSTM. While showing promising results, this model still faces two challenges. **First**, caused by the extension of the already slow LSTM to DAG structured model, lattice LSTM is restricted to processes one data at a time, which means it is seriously inefficient and difficult to deploy. **Second**, due to the inherently unidirectional sequential nature, lattice LSTM fails to incorporate the semantics of a lexical word into the representation of the characters it encompasses except the last one, while such information is crucial for character-level sequence tagging. Taking the sentence in Figure 1 as an example, lattice LSTM directly flows the information of the lexical word "南京市(NanJing City)" to "市(City)" and skips the other two inside characters "南(South)" and "京(Capital)", but the semantics and boundary information of "南京市(NanJing City)" can be the key knowledge to predict the tag of "南(South)" as "B-LOC".

In this paper, we consider all these issues systematically and present a novel **P**orous **L**attice **T**ransformer **E**ncoder (PLTE) to deal with these issues. Inspired by previous research on translation (Xiao et al., 2019; Sperber et al., 2019) that integrated lattice-structured inputs into self-attention models, we propose the lattice transformer encoder for Chinese NER by introducing lattice-aware self-attention, which borrows the idea from the relative positional embedding (Shaw et al., 2018) to make self-attention aware of the relative position information in lattice structure. Considering that self-attention network calculates attention weights between each pair of tokens in a sequence regardless of their distance, we can simply concatenate all the characters and lexical words as input to consume lattices without resorting to the DAG structure. In this way, characters coupled with lexical words can be processed in batches. Meanwhile, a lexical word representation is allowed to build a direct relation with the included characters by lattice-aware self-attention, thus addressing the second issue. Additionally, some pioneering works (Yang et al., 2018, 2019) demonstrate that self-attention benefits from locality modeling. Especially in NER task, the local compositionality is indeed illuminating. As we can see from the example in Figure 1, the word "市长(Mayor)" is the immediate and most obvious sign to guide the neighboring character "江(River)" to be identified as "B-PER" instead of "B-LOC". To this end, we further introduce a novel porous mechanism to enhance the local dependencies among neighboring tokens. The key insight is to sparse the self-attention architecture by replacing the fully-connected topology with a pivot-shared structure. In this manner, every two non-neighboring tokens are connected by a shared pivot node, and we strengthen the dependency for two neighboring tokens by connecting them directly.

In summary, this paper makes the following contributions: (1) We propose a novel lattice transformer encoder for Chinese NER, which is capable of handling lattices in batch mode and capturing dependencies between characters and matched lexical words. (2) We revise lattice-aware attention distribution via a porous mechanism, which enhances the ability of capturing useful local context. (3) Experimental results on four datasets demonstrate that our model performs up to 11.4 times faster than baseline methods and achieves better performance. Furthermore, we show that our model can be easily integrated into the pre-trained language model BERT (Devlin et al., 2019), and combining them further improves the state of the art.

2 Background

In this section, we first briefly review self-attention mechanism, then move on to current lattice Transformer that our PLTE model is built upon.

2.1 Self-Attention

Self-attention mechanism has attracted increasing attention due to their flexibility in parallel computation and dependency modeling. Given an input sequence representation $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\} \in \mathbb{R}^{n \times d}$, we can first transform it into queries $\mathbf{Q} = \mathbf{X}\mathbf{W}^Q \in \mathbb{R}^{n \times d_k}$, keys $\mathbf{K} = \mathbf{X}\mathbf{W}^K \in \mathbb{R}^{n \times d_k}$, and values $\mathbf{V} = \mathbf{X}\mathbf{W}^V \in \mathbb{R}^{n \times d_v}$, where $\{\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V\}$ are trainable parameters. The output sequence representation is calculated as:

$$\mathrm{Att}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{Softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}})\mathbf{V}, \qquad (1)$$

where $\sqrt{d_k}$ is the scaling factor.

2.2 Lattice Transformer

Transformer has achieved great success on many NLP tasks, notably machine translation and language modeling. By invoking multi-layer self-attention for global context modeling, Transformer enables paralleled computation and addresses the inherent sequential computation shortcoming of RNNs. Lattice Transformer is a generalization of the standard transformer architecture to accept lattice-structured inputs. Specifically, it linearizes the lattice structure and introduces a position relation score matrix to make self-attention aware of the topological structure of lattice:

$$\mathrm{Att}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{Softmax}(\frac{\mathbf{Q}\mathbf{K}^T + \mathbf{R}}{\sqrt{d_k}})\mathbf{V}, \qquad (2)$$

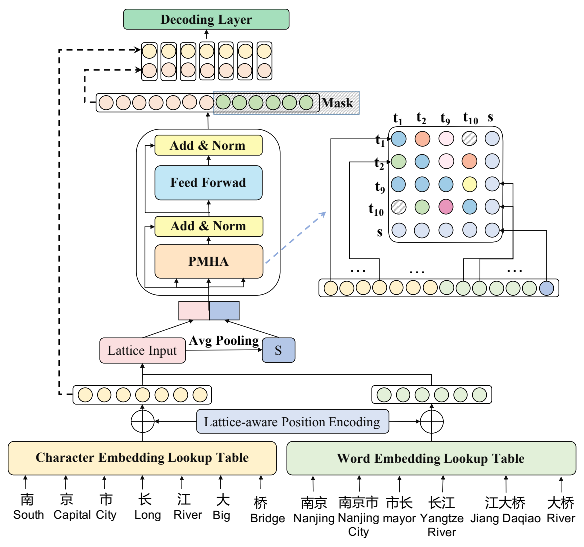


Figure 2: The overall architecture of PLTE (best viewed in color). Characters and lexical words are shown in yellow and green, respectively. We concatenate character embeddings and word embeddings as lattice input. When decoding, we mask words and just make sequence labeling for characters.

where $\mathbf{R} \in \mathbb{R}^{n \times n}$ encodes the lattice-dependent relations between each pair of elements from the lattice input, and its computational method depends on the specific definition of the relations according to the task objective.

3 Models

3.1 Lattice Input Layer

The input layer aims to embed both semantic information and position information of tokens into their token embeddings.

Word-Character Embedding Formally, let $S = \{c_1, ..., c_M\}$ denotes a sentence, where c_i is the i-th character. The lexical words in the lexicon that match a character subsequence can be formulated as $e_{i:j}$, where the index of the first and last letters are i and j, respectively. Similarly, we can also represent c_i as $e_{i:i}$. As shown in the top half of Figure 3, $e_{3:4}$ indicates the lexical word named "市长(Mayor)" which contains c_3 named "市(City)" and c_4 named "长(Long)". Each character c_i can be turned into the vector $\mathbf{x}_i^c$ which includes it's character embedding and bigram embedding. By looking up the vector from a pre-trained word embedding matrix, each matched lexical word $e_{i:j}$ is represented as a vector $\mathbf{x}_{i:j}^w$.

Lattice-Aware Position Encoding Since self-attention architecture contains no recurrence, to make the model aware of the sequence order, we add position embedding to the semantic embedding of each token. Specifically, the position of a character is defined as its absolute position in the input sequence S. And the position of a matched word is the position of its first character. For example, in Figure 3, the position of word "南京(Nanjing)" is 1 because this sentence begins with "南(South)".

Finally, since position information is incorporated into token embeddings, we can simply put the matched words to the end of the character sequence S and form a new token sequence $T = \{t_i\}_{i=1}^N$ to consume lattice structure, where N is the sum of the number of characters and words. See the top half of Figure 3 for the detailed correspondence.

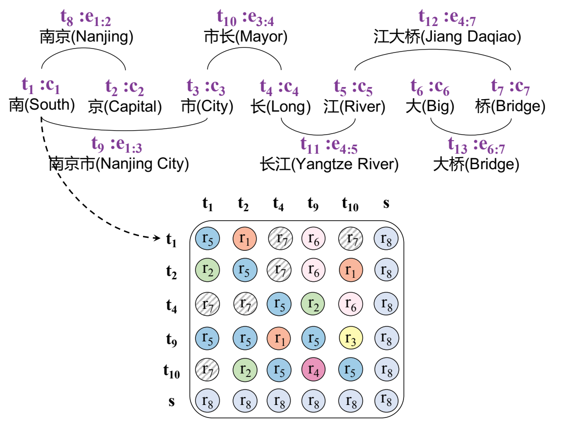


Figure 3: Illustration of the relative position relation matrix. Notice that we present several relations among partial tokens as instances. Different colors indicate different relations defined in Figure 4. For instance, the relation between t_4 and t_{10} is r_6, since that "长(Long)" is included in "市长(Mayor)". The circle filled with lines denotes that we don't compute attention between non-neighboring tokens due to our porous mechanism.

3.2 Porous Lattice Transformer Encoder

As mentioned in the Introduction, our primary goal is to adapt the standard transformer to the task of Chinese NER with lattice inputs. To this end, we first propose lattice-aware self-attention to consume input tokens and the relative position information of lattice structure. Then, we design a porous mechanism which learns sparse attention coefficients by replacing the fully-connected topology with a pivot-shared structure to enhance the association between neighboring elements. Besides, we also use multi-head attention (Vaswani et al., 2017) to capture information from different representation subspaces jointly.

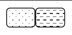
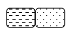
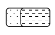
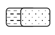
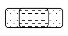
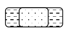
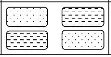
	Conditions		Relations
r_1	$q = k - 1$		$e_{p:q}$is left adjacent to $e_{k:l}$
r_2	$l = p - 1$		$e_{p:q}$ is right adjacent to $e_{k:l}$
r_3	$p < k \le q < l$		$e_{p:q}$is left intersected with $e_{k:l}$
r_4	$k < p \le l < q$		$e_{p:q}$is right intersected with $e_{k:l}$
r_5	$p \le k \le l \le q$		$e_{p:q}$ includes $e_{k:l}$
r_6	$k \le p \le q \le l$		$e_{p:q}$ is included in $e_{k:l}$
r_7	$q < k - 1$ *or* $l < p - 1$		$e_{p:q}$ is non-neighboring to $e_{k:l}$

Figure 4: Relations between $e_{p:q}$ and $e_{k:l}$. We use the block filled with dots and lines to present $e_{p:q}$ and $e_{k:l}$, respectively. Notice that if $p = q = k = l$, we denote the relation between $e_{p:q}$ and $e_{k:l}$ as r_5.

Lattice-Aware Self-Attention (LASA) Original position embedding described above only indicates the sequential order and cannot capture the relative position information of the lattice-structured input. For example, in Figure 3, the sequential distance from "市(City)" or "市长(Mayor)" to "长(Long)' is 1 under previous position definition. Actually, "长(Long)" is included in "市长(Mayor)" and right adjacent to "市(City)", but absolute position fails to make a distinction. To address this issue, we propose a relative position relation matrix $L \in \mathbb{N}^{N\times N}$ to present such position information. Similar to (Xiao et al., 2019), we enumerate all possible relations between each pair of elements $e_{p:q}$ and $e_{k:l}$ in Figure 4. We give a detailed and vivid example in Figure 3. For two tokens t_i and t_j refering to $e_{p:q}$ and $e_{k:l}$ respectively, the matrix entry $L_{i,j}$ is the pre-defined relation between them, such as $L_{1,2} = r_1$.

More concretely, in order to make L learnable, we first represent L as the relation position embedding, a 3D tensor $\mathbf{R} \in \mathbb{R}^{N\times N\times d_r}$ by looking up a trainable embedding matrix $\mathbf{A} \in \mathbb{R}^{8\times d_r}$, where d_r is the relational embedding dimensionality. Note that here we define eight types of embedding instead of seven relations in Figure 4. The additional embedding is introduced to represent the interaction relation with a shared pivot node (described in the next section) and facilitate parallel computation. Then, to incorporate such position relations into attention layer, we adapt Equation 2 as follows:

$$\alpha = \mathrm{Softmax}\Big(\frac{\mathbf{Q}\mathbf{K}^T + \textbf{einsum}(\text{“}ik,ijk \to ij\text{”}, \mathbf{Q}, \mathbf{R}^K)}{\sqrt{d_k}}\Big) \quad (3)$$

$$\mathrm{Att}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \alpha\mathbf{V} + \textbf{einsum}(\text{“}ik, ikj \to ij\text{”}, \alpha, \mathbf{R}^V) \quad (4)$$

where $\mathbf{R}^K \in \mathbb{R}^{N\times N\times d_k}$ and $\mathbf{R}^V \in \mathbb{R}^{N\times N\times d_v}$ are two relation embedding tensors which are added to the keys and values respectively to indicate relation between input tokens. In our case, $\mathbf{Q}$ is a 2D array of shape $[N \times d_k]$ while $\mathbf{R}^K$ is a 3D array and we need to result in a new array of shape $[N \times N]$, with the element in i-th row and j-th column is $\sum_{k=1}^{d_k} \mathbf{Q}_{ik}\mathbf{R}^K_{ijk}$. To implement this operation, we apply **einsum**[2] to sum out the dimension of the hidden size, where einsum is an operation computing multilinear expressions (i.e., sums of products) using the Einstein summation convention.

Porous Multi-Head Attention (PMHA) Considering standard self-attention mechanism encodes sequences by relating sequence items to another one through computation of pairwise similarity, it disperses the distribution of attention and overlooks the local knowledge provided by neighboring elements, which is crucial for NER. To maintain the strength of capturing long distance dependencies and enhance the ability of capturing short-range dependencies, we sparsify the transformer architecture by replacing the fully-connected topology with a pivot-shared structure referenced by (Guo et al., 2019). Specifically, given element set E and its embedding matrix $\mathbf{X}$, where $e_{i:j} \in E$ and $\mathbf{x}_{i:j} \in \mathbf{X}$ (if $e_{i:j}$ is a character then $\mathbf{x}_{i:j} = \mathbf{x}^c_i$ else $\mathbf{x}_{i:j} = \mathbf{x}^w_{i:j}$), we define $e^{r_k}_{i:j}$ as the element set whose relation with $e_{i:j}$ is r_k, $\mathbf{x}^{r_k}_{i:j}$ as the concatenation of the embeddings where each embedding represents the corresponding element in $e^{r_k}_{i:j}$. we also define the neighboring set of $e_{i:j}$ as $\varepsilon = \{e^{r_1}_{i:j}; e^{r_2}_{i:j}; e^{r_3}_{i:j}; e^{r_4}_{i:j}; e^{r_5}_{i:j}; e^{r_6}_{i:j}\}$, then we update the hidden state $\mathbf{h}_{i:j}$ of $e_{i:j}$ with multi-head attention as follows:

$$\begin{aligned}
\mathbf{h}_{i:j} &= [\mathbf{z}^1_{i:j}; \mathbf{z}^2_{i:j}; ...; \mathbf{z}^H_{i:j}]\mathbf{W}^O \\
\mathbf{z}^h_{i:j} &= \mathrm{Att}(\mathbf{x}_{i:j}\mathbf{W}^Q_h, \mathbf{c}_{i:j}\mathbf{W}^K_h, \mathbf{c}_{i:j}\mathbf{W}^V_h),\ h \in [1, H] \\
\mathbf{c}_{i:j} &= [\mathbf{x}^{r_1}_{i:j}; \mathbf{x}^{r_2}_{i:j}; \mathbf{x}^{r_3}_{i:j}; \mathbf{x}^{r_4}_{i:j}; \mathbf{x}^{r_5}_{i:j}; \mathbf{x}^{r_6}_{i:j}; \mathbf{s}] \qquad (5) \\
\mathbf{s} &= \frac{1}{n}\sum_{i,j} \mathbf{x}_{i:j},
\end{aligned}$$

where $\mathbf{W}^Q_h, \mathbf{W}^K_h, \mathbf{W}^V_h$ are trainable projection matrices corresponding to the h-th head, $\mathbf{z}_h$ is the h-th output, H is the number of heads and Att() is defined in Equation 4. As we can see, in our porous multi-head attention, one element $e_{i:j}$ just makes direct attention computation with its neighboring elements and models the non-local compositions via the pivot node $\mathbf{s}$. As illustrated in Figure 3, $e_{i:j}$ doesn't compute attention directly with the element

[2]This operation is available in Numpy, TensorFlow, and Pytorch.

set $e^{r7}_{i:j}$, thus we mask them. Under this lightweight porous structure, our transformer encoder has an approximate ability to strengthen local dependencies among neighboring tokens and keep the ability to capture long distance dependencies.

3.3 BiGRU-CRF Decoding

After extracting the semantic information by the porous lattice transformer encoder layer, we feed the character sequence representations into a BiGRU-CRF decoding layer to make sequence tagging. Specifically, taking $[\mathbf{x}^c_1; \mathbf{h}_{1:1}], ..., [\mathbf{x}^c_n; \mathbf{h}_{n:n}]$ as input, a bidirectional GRU is implemented to produce forward state $\overrightarrow{\mathbf{h}}_t$ and backward state $\overleftarrow{\mathbf{h}}_t$ for each time step, and then we concatenate these two separate hidden states as the encoding output of the t-th character, donated as $\mathbf{h}_t = [\overrightarrow{\mathbf{h}}_t; \overleftarrow{\mathbf{h}}_t]$.

Finally, a standard CRF layer is used on top of $\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_n$ to make sequence tagging. For a label sequence $\mathbf{y} = \{y_1, y_2, ..., y_n\}$, we define its probability to be:

$$P(\mathbf{y}|S) = \frac{\exp(\sum_i (\mathbf{W}^{y_i}_{\text{CRF}} \mathbf{h}_i + b^{(y_{i-1}, y_i)}_{\text{CRF}}))}{\sum_{y'} \exp(\sum_i (\mathbf{W}^{y'_i}_{\text{CRF}} \mathbf{h}_i + b^{(y'_{i-1}, y'_i)}_{\text{CRF}}))} \qquad (6)$$

Here y' denotes all possible tag sequences, $\mathbf{W}^{y_i}_{\text{CRF}}$ is a model parameter specific to y_i, and $b^{(y_{i-1}, y_i)}_{\text{CRF}}$ is the transition score between y_{i-1} and y_i. While decoding, we use the first-order Viterbi algorithm to find the label sequence that obtains highest score.

3.4 Training

Given a set of manually labeled training data $\{(S_i, \mathbf{y}_i)\}|^N_{i=1}$. sentence-level log-likelihood loss with L_2 regularization is used to train the model:

$$\mathcal{L} = \sum_{i=1}^{N} \log(P(\mathbf{y}_i|S_i)) + \frac{\lambda}{2} \parallel \Theta \parallel^2, \qquad (7)$$

where λ is the L_2 regularization weight and Θ represents the parameter set.

4 Experiments

4.1 Experimental Setup

4.1.1 Data

We evaluate our proposed model on four datasets, including OntoNotes (Ralph et al., 2011), MSRA (Levow, 2006), Weibo NER (Peng and Dredze, 2015; He and Sun, 2017b) and a Chinese Resume dataset (Zhang and Yang, 2018). We use the same training, development, and test split as (Zhang and Yang, 2018). For these four datasets, both OntoNotes and MSRA are in news domain, while Weibo and Resume NER come from social media.

4.1.2 Comparison Methods

We first compare our proposed model to several recent lexicon-based models.

Lattice LSTM. Lattice LSTM (Zhang and Yang, 2018) exploits lexical information in character sequence through gated recurrent cells, which can avoid segmentation errors.

LR-CNN. LR-CNN (Gui et al., 2019a) is the latest SOTA method of Chinese NER, which incorporates lexicons using a rethinking mechanism.

Furthermore, to explore the effectiveness of pre-trained language model, we implement several baselines based on BERT representations.

BERT-Tagger. BERT-Tagger (Devlin et al., 2019) uses the outputs from the last layer of model BERT$_{base}$ as the character-level enriched contextual representations to make sequence labelling.

PLTE[BERT]/LR-CNN[BERT]/Lattice LSTM[BERT]. These three models replace character embedding layer with the pre-trained BERT representations, and use softmax layer to make sequence tagging.

4.1.3 Hyper-parameter settings

In our experiments, we use the same word and character embeddings as (Zhang and Yang, 2018), which are pre-trained on Chinese Giga-word [3] using Word2vec (Mikolov et al., 2013) and fine-tuned during training. The model is trained using stochastic gradient descent with the initial learning rate of 0.045 and the weight decay of 0.05. Dropout is applied to the embeddings and GRU layer with a rate of 0.5 and the transformer encoder with 0.3. For the biggest dataset MSRA and the smallest dataset Weibo, we set the dimensionality of GRU hidden states as 200 and 80 respectively. For the other datasets, this dimension is set to 100. What's more, the hidden size and the number of heads are set to 128 and 6, respectively. For models based on BERT, we fine-tune BERT representation layer during training. We use BertAdam to optimize all the trainable parameters, select the best learning rate from $1e$-5 to $1e$-4 on the development set.

4.2 Experimental Results

OntoNotes. Table 1 illustrates our experimental results on OntoNotes. The "Input" column indi-

[3] `https://catalog.ldc.upenn.edu/LDC2011T13`

Input	Models	P	R	F1
Gold seg	Che et al. (2013)	77.71	72.51	75.02
	Wang et al. (2013)	76.43	72.32	74.32
	Yang et al. (2016)	72.98	80.15	**76.40**
No seg	Lattice LSTM (2018)	76.35	71.56	73.88
	LR-CNN (2019a)	76.40	72.60	74.45
	CAN-NER(2019)	75.05	72.29	73.64
	PLTE	76.78	72.54	**74.60**
	BERT-Tagger	78.01	80.35	79.16
	Lattice LSTM[BERT]	79.79	79.41	79.60
	LR-CNN[BERT]	79.41	80.32	79.86
	PLTE[BERT]	79.62	81.82	**80.60**

Table 1: Main results on OntoNotes.

Models	P	R	F1
Zhou et al. (2013)	91.86	88.75	90.28
Lu et al. (2016)	-	-	87.94
Cao et al. (2018)	91.73	89.58	90.64
Lattice LSTM (2018)	93.57	92.79	93.18
CAN-NER(2019)	93.53	92.42	92.97
LR-CNN (2019a)	94.50	92.93	**93.71**
PLTE	94.25	92.30	93.26
BERT-Tagger	94.43	93.86	94.14
Lattice LSTM[BERT]	93.99	92.86	93.42
LR-CNN[BERT]	94.68	94.03	94.35
PLTE[BERT]	94.91	94.15	**94.53**

Table 2: Main results on MSRA.

Models	P	R	F1
Peng and Dredze (2016)	66.47	47.22	55.28
He and Sun (2017a)	61.68	48.82	54.50
Cao et al. (2018)	59.51	50.00	54.43
Lattice LSTM (2018)	52.71	53.92	53.13
LR-CNN (2019a)	65.06	50.00	**56.54**
PLTE	62.21	49.54	55.15
BERT-Tagger	67.12	66.88	67.33
Lattice LSTM[BERT]	61.08	47.22	53.26
LR-CNN[BERT]	64.11	67.77	65.89
PLTE[BERT]	72.00	66.67	**69.23**

Table 3: Main results on Weibo NER.

Models	P	R	F1
Lattice LSTM (2018)	94.81	94.11	94.46
CAN-NER(2019)	95.05	94.82	94.94
LR-CNN (2019a)	95.37	94.84	95.11
PLTE	95.34	95.46	**95.40**
BERT-Tagger	96.12	95.45	95.78
Lattice LSTM[BERT]	95.79	95.03	95.41
LR-CNN[BERT]	95.68	96.44	96.06
PLTE[BERT]	96.16	96.75	**96.45**

Table 4: Main results on Resume NER.

cates that the input sentences are segmented or not, where methods in **Gold seg** process word sequences with gold segmentation and **No seg** indicates that whether the input sentence is a character sequence which has not been segmented.

With gold-standard segmentation, all of these word-level models (Che et al., 2013; Wang et al., 2013; Yang et al., 2016) achieve great performance by using segmentation and external labeled data. But such information is not available in most datasets, such that we only use pre-trained character and word embeddings as our resource.

In No-segmentation settings, we first present 3 non-BERT models which are widely used. Our PLTE model achieves the best F1 score and gains a 0.72% improvement over lattice LSTM in F1 score since our model integrates lexical words information into self-attention computation in a more effective fashion.[4] While with pre-trained BERT$_{base}$ model, the BERT-Tagger leads to a significant jump in performance to 79.16%. On the basis of the great success using BERT representations, our proposed PLTE[BERT] model outperforms the BERT-Tagger by 1.44% in F1 score on OntoNotes.

MSRA/Weibo/Resume Tables 2, 3, and 4 present the comparisons among various methods on the MSRA, Weibo, and Resume datasets. Existing statistical methods explore the rich statistical features (Zhou et al., 2013) and character embedding features (Lu et al., 2016). For neural based models, some existing models utilize multi-task learning (Peng and Dredze, 2016; Cao et al., 2018) or semi-supervised learning (He and Sun, 2017a). CAN-NER(Zhu and Wang, 2019) investigate a character-based convolutional attention network coupled with GRU for Chinese NER.

Consistent with observations on OntoNotes, all lexicon-based methods achieve higher F1 scores than the character-based methods, which demonstrates the usefulness of lexical word information. With pre-trained contextual representations, BERT-based models outperform non-BERT models by a large margin. Even though the original BERT model already provides strong prediction power, PLTE consistently improves over BERT-Tagger, lattice LSTM[BERT] and LR-CNN[BERT], which indicates that our proposed PLTE model can better utilize these semantic representations. Another interesting observation is that PLTE gains more significant improvement when combined with BERT compared with other lexicon-based methods. We suspect it is because PLTE is indeed effective than baseline methods and thus more capable of fully leveraging the language information embedded in

[4]Case study is provided in Table 1 in the Appendix.

Models	Word2vec			BERT		
	Lattice LSTM	LR-CNN	PLTE	Lattice LSTM	LR-CNN	PLTE
OntoNotes	1×	2.23×	11.4×	1×	1.96×	6.21×
MSRA	1×	1.57×	8.48×	1×	1.97×	7.11×
Weibo	1×	2.41×	9.12×	1×	2.02×	6.48×
Resume	1×	1.44×	9.68×	1×	1.46×	5.57×

Table 5: Testing-time speedup of different models. Lattice LSTM and LR-CNN can only run with $batch\_size = 1$ while our PLTE model runs with $batch\_size = 16$.

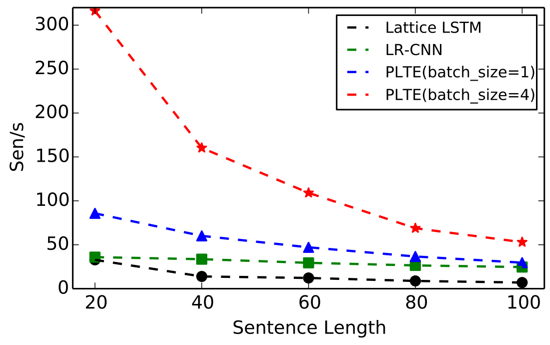


Figure 5: Test speed against sentence length. Sen/s denotes the number of sentences processed per second.

the input representations. While the embeddings pre-trained by Word2vec are not as informative as enough to allow PLTE to fulfill its potential, BERT representation can well capture rich semantic patterns and help PLTE exploit the performance.

4.3 Experimental Discussion

4.3.1 Efficiency Advantage

PLTE not only achieves excellent performance, but also outperforms current lexicon-based methods in efficiency. Table 5 lists the test time of different models with different input representations on all four benchmarks. As we can see, PLTE decodes up to 11.4 and 5.11 times faster than lattice LSTM and LR-CNN respectively with Word2vec embeddings on OneNotes. Similar efficiency improvement can also be observed on other datasets under both two kinds of input representations. Actually, aligning word-character lattice structure for batch running is usually non-trivial (Sui et al., 2019) and both lattice LSTM and LR-CNN have no ability in batch-running due to the DAG structure or variable-sized lexical words set, while PLTE overcomes this limitation since we can simply concatenate all the elements as input to consume lattices owing to the lattice-aware self-attention mechanism, which calculates attention weights between each pair of tokens by matrix multiplication, thus can be computed parallelly in batches.

Models	OntoNotes	MSRA	Weibo	Resume
PLTE	74.60	93.26	59.76	95.40
-LASA	70.97	92.27	56.95	94.82
-PM	70.17	91.84	50.08	94.40
-LASA-PM	70.58	92.48	56.52	94.69

Table 6: An ablation study of our proposed model. For model without lattice-aware self-attention (-LASA), we take character sequence as input, and one character just computes multi-head self-attention weights with its adjacent characters and the shared pivot node. For model without porous mechanism (-PM), we directly utilize multi-head LASA to aggregate the weighted information of each word with fully-connected attention connections. For PLTE-LASA-PMHA, we apply multi-head self-attention to each pair of elements from the input character sequence.

To investigate the influence of the different sentence lengths, we conduct experiments on OntoNotes by splitting this dataset into five parts according to sentence length. The results in Figure 5 demonstrate that PLTE performs faster than lattice LSTM and LR-CNN with different sentence lengths, especially for short sentences. In particular, when the sentence length is less than 20 characters, PLTE($batch\_size = 4$) performs 9.64 times faster than lattice LSTM and 8.81 times faster than LR-CNN. With the sentence length increases, the efficiency gains from batching computation decline gradually due to the limited computing resources of a single GPU. Besides, even if the batch size is set to 1, PLTE still has remarkable advantage in speed compared with baselines, since lattice LSTM demands multiple recurrent computation, and the rethinking mechanism in LR-CNN is also computationally expensive, which again demonstrates our strength in terms of efficiency.

4.3.2 Model Ablation study

To demonstrate the effectiveness of each component, we conduct an ablation study on four datasets. From the results in Table 6, we can observe that: (1) Removing the LASA module hurts the results by 3.63%, 0.99%, 2.81% and 0.58% F1 score on four datasets respectively, which indicates that lexicons play an important role in character-level Chinese NER. (2) By introducing the porous mechanism (PM), we can enhance the ability of capturing useful local context which is beneficial to NER, while maintaining the strength of capturing long-term dependencies. (3) PLTE-PM performs worse than PLTE-LASA-PM, which confirms that the standard LASA is not suitable for NER because it takes into account all the signals and disperses the distribution of attention, while NER may be benefited more from localness modeling. (4) PLTE-LASA outperforms PLTE-LASA-PM on most datasets, which shows that the porous mechanism can also benefit self-attention when only taking characters as input.

5 Related Work

In this section, we summarize the NER models based on neural networks and lattice transformer models which our work is in line with.

Huang et al. (2015) proposed a BiLSTM-CRF model for NER and achieved good performance. Santos and Guimaraes (2015) utilized word- and character-level representations based on the CharWNN deep neural network. Lample et al. (2016) designed a character LSTM and word LSTM for NER. But these word-based methods always suffer from segmentation errors, resulting in an unsatisfactory performance.

To avoid the segmentation errors, most recent NER models are built upon character sequence labeling. Peng and Dredze (2015) proposesd a joint training objective for three types of neural embeddings to better recognize entity boundary. Lu et al. (2016) presented a position-sensitive skip-gram model to learn multi-prototype Chinese character embeddings. He and Sun (2017a) took the positional character embeddings into account. Although these methods achieve promising performance, they all ignore the word information lying in character sequence.

Some researchers then devoted themselves to exploiting rich word boundary and semantic information in character sequence. Cao et al. (2018) applied an adversarial transfer learning framework to integrate the task-shared word boundary information into Chinese NER. Liu et al. (2019) explored four different strategies for Word-Character LSTM. Gui et al. (2019a) proposed a CNN-based NER model that incorporates lexicons using a rethinking mechanism. Especially, recent SOTA methods exploit lattice-structured models to integrate latent word information into character sequence, which has been proven effective on various NLP tasks (Su et al., 2017; Tan et al., 2018) . Specifically, Zhang and Yang (2018) utilized the lattice LSTM to leverage explicit word information over character sequence labeling. Based on this method, Gui et al. (2019b) and Sui et al. (2019) formulated the lattice structure as a graph and leveraged Graph Neural Networks (GNNs) to integrate lexical knowledge. However, for NER task, coupling pre-trained language models with GNNs and fine-tuning them is usually non-trivial.

Lattice transformer has been exploited in NMT (Xiao et al., 2019), as well as speech translation (Sperber et al., 2019; Zhang et al., 2019). Compared with existing work, our proposed porous lattice transformer encoder is different in both motivation and structure. More specifically, we remove the decoder architecture and reduce the number of the encoder layers. Besides, we revise the fully-connected attention distribution with a pivot-shared structure via the porous mechanism to enhance the local dependencies among neighboring tokens.[5] To our knowledge, we are the first to design a novel porous lattice transformer encoder for segmentation-free Chinese NER.

6 Conclusions

In this paper, we present PLTE, a novel porous lattice transformer encoder which incorporates lexicons into character-level Chinese NER. PLTE enables the interaction between the matched lexical words and their constituent characters and proceeds in batches with the lattice-aware self-attention. It also learns a porous attention distribution to enhance the ability of localness modeling. We evaluate the proposed model on four Chinese NER datasets. Using Word2vec embeddings, our approach PLTE outperforms various baseline models and performs up to 11.4 times faster than previous lattice-based method. After switching to BERT representations, we show that PLTE achieves more significant performance gain than other methods. There are multiple venues for future work. One promising direction is to apply our model to the pre-training procedure of language models.

[5]Differences between lattice self-attention and porous lattice self-attention are shown in Figure 1 in the Appendix.

References

Adam Berger and John Lafferty. 2017. Information Retrieval as Statistical Translation. In *SIGIR Forum*, pages 219–226.

Pengfei Cao, Yubo Chen, Kang Liu, Jun Zhao, and Shengping Liu. 2018. Adversarial Transfer Learning for Chinese Named Entity Recognition with Self-Attention Mechanism. In *EMNLP*, pages 182–192.

Wanxiang Che, Mengqiu Wang, Christopher D. Manning, and Ting Liu. 2013. Named Entity Recognition with Bilingual Constraints. In *HLT-NAACL*, pages 52–62.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186.

Tao Gui, Ruotian Ma, Qi Zhang, Lujun Zhao, Yu-Gang Jiang, and Xuanjing Huang. 2019a. CNN-Based Chinese NER with Lexicon Rethinking. In *IJCAI*, pages 4982–4988.

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 *EMNLP*, pages 1039–1049.

Qipeng Guo, Xipeng Qiu, Pengfei Liu, Yunfan Shao, Xiangyang Xue, and Zheng Zhang. 2019. Star-Transformer. In *NAACL*, pages 1315–1325.

Hangfeng He and Xu Sun. 2017a. A Unified Model for Cross-Domain and Semi-Supervised Named Entity Recognition in Chinese Social Media. In *AAAI*, pages 3216–3222.

Hangfeng He and Xu Sun. 2017b. F-Score Driven Max Margin Neural Network for Named Entity Recognition in Chinese Social Media. In *EACL*, pages 713–718.

Zhiheng Huang, Wei Liang Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF Models for Sequence Tagging. *arXiv: Computation and Language*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. In *NAACL:HLT*, pages 260–270.

Gina-Anne Levow. 2006. The third international Chinese language processing bakeoff: Word segmentation and named entity recognition. In *In Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 108–117.

Wei Liu, Tongge Xu, Qinghua Xu, Jiayu Song, and Yueran Zu. 2019. An Encoding Strategy Based Word-Character LSTM for Chinese NER. In *NAACL*, pages 2379–2389.

Yanan Lu, Yue Zhang, and Dong-Hong Ji. 2016. Multi-prototype Chinese Character Embedding. In *LREC*, pages 855–859.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Nanyun Peng and Mark Dredze. 2015. Named Entity Recognition for Chinese Social Media with Jointly Trained Embeddings. In *EMNLP*, pages 548–554.

Nanyun Peng and Mark Dredze. 2016. Improving Named Entity Recognition for Chinese Social Media with Word Segmentation Representation Learning. In *ACL*, pages 149–155.

Weischedel Ralph, Pradhan Sameer, Ramshaw Lance, Palmer Martha, Xue Nianwen, Marcus Mitchell, Taylor Ann, Greenberg Craig, Hovy Eduard, Belvin Robert, and et al. 2011. Ontonotes release 4.0. In *LDC2011T03, Philadelphia, Penn.: Linguistic Data Consortium*.

Cicero Nogueira Dos Santos and Victor Guimaraes. 2015. Boosting Named Entity Recognition with Neural Character Embeddings. In *Proceedings of the Fifth Named Entity Workshop*, pages 25–33.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *NAACL*, pages 464–468.

Matthias Sperber, Graham Neubig, Ngoc-Quan Pham, and Alex Waibel. 2019. Self-Attentional Models for Lattice Inputs. In *ACL*, pages 1185–1197.

Jinsong Su, Zhixing Tan, Deyi Xiong, Rongrong Ji, Xiaodong Shi, and Yang Liu. 2017. Lattice-based recurrent neural network encoders for neural machine translation. In *AAAI*, pages 3302–3308.

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 *EMNLP*, pages 3821–3831.

Zhixing Tan, Jinsong Su, Boli Wang, Yidong Chen, and Xiaodong Shi. 2018. Lattice-to-sequence attentional neural machine translation models. In *Neurocomputing*, pages 138–147.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NIPS*, pages 5998–6008.

Mengqiu Wang, Wanxiang Che, and Christopher D. Manning. 2013. Effective Bilingual Constraints for Semi-supervised Learning of Named Entity Recognizers. In *AAAI*, pages 919–925.

Fangzhao Wu, Junxin Liu, Chuhan Wu, Yongfeng Huang, and Xing Xie. 2019. Neural Chinese Named Entity Recognition via CNN-LSTM-CRF and Joint Training with Word Segmentation. In *WWW*, pages 3342–3348.

Fengshun Xiao, Jiangtong Li, Hai Zhao, Rui Wang, and Kehai Chen. 2019. Lattice-Based Transformer Encoder for Neural Machine Translation. In *ACL*, pages 3090–3097.

Mengge Xue, Weiming Cai, Jinsong Su, Linfeng Song, Yubin Ge, Yubao Liu, and Bin Wang. 2019. Neural Collective Entity Linking Based on Recurrent Random Walk Network Learning. In *IJCAI*, pages 5327–5333.

Baosong Yang, Zhaopeng Tu, Derek F. Wong, Fandong Meng, Lidia S. Chao, and Tong Zhang. 2018. Modeling Localness for Self-Attention Networks. In *EMNLP*, pages 4449–4458.

Baosong Yang, Longyue Wang, Derek F. Wong, Lidia S. Chao, and Zhaopeng Tu. 2019. Convolutional Self-Attention Networks. In *NAACL*, pages 4040–4045.

Jie Yang, Zhiyang Teng, Meishan Zhang, and Yue Zhang. 2016. Combining Discrete and Neural Features for Sequence Labeling. In *CICLing*, pages 919–925.

Bowen Yu, Zhenyu Zhang, Tingwen Liu, Bin Wang, Sujian Li, and Quangang Li. 2019. Beyond Word Attention: Using Segment Attention in Neural Relation Extraction. In *IJCAI*, pages 5401–5407.

Pei Zhang, Niyu Ge, Boxing Chen, and Kai Fan. 2019. Lattice Transformer for Speech Translation. In *ACL*, pages 6475–6484.

Yue Zhang and Jie Yang. 2018. Chinese NER Using Lattice LSTM. In *ACL*, pages 1554–1564.

J. Zhou, W. Qu, and F. Zhang. 2013. Chinese Named Entity Recognition via Joint Identification and Categorization. *Chinese Journal of Electronics*, pages 225–230.

Yuying Zhu and Guoxin Wang. 2019. CAN-NER: Convolutional Attention Network for Chinese Named Entity Recognition. In *NAACL*, pages 3384–3393.