Lattice LSTM for Chinese Sentence Representation

Yue Zhang, Yile Wang, and Jie Yang

***Abstract*—Words provide a useful source of information for Chinese NLP, and word segmentation has been taken as a pre-processing step for most downstream tasks. For many NLP tasks, however, word segmentation can introduce noise and lead to error propagation. The rise of neural representation learning models allows sentence-level semantic information to be collected from characters directly. As a result, it is an empirical question whether a fully character-based model should be used instead of first performing word segmentation. We investigate a neural representation that simultaneously encodes character and word information without the need for segmentation. In particular, candidate words are found in a sentence by matching with a pre-defined lexicon. A lattice structured LSTM is used to encode the resulting word-character lattice, where gate vectors are used to control information flow through words, so that the more useful words can be automatically identified by end-to-end training. We compare the performance of the resulting lattice LSTM and baseline sequence LSTM structures over both character sequences and automatically segmented word sequences. Results on NER show that the character-word lattice model can significantly improve the performance. In addition, as a general sentence representation architecture, character-word lattice LSTM can also be used for learning contextualized representations. To this end, we compare lattice LSTM structure with its sequential LSTM counterpart, namely ELMo. Results show that our lattice version of ELMo gives better language modeling performances. On Chinese POS-tagging, chunking and syntactic parsing tasks, the resulting contextualized Chinese embeddings also give better performance than ELMo trained on the same data.**

***Index Terms*—Lattice LSTM, NER, language modeling, contextualize representation.**

I. Introduction

CHINESE sentences are naturally written as sequences of characters. On the other hand, words are a basic unit of semantic information, and have been taken as a basic source of features for Chinese NLP. As a result, word segmentation [1] has been taken as a pre-processing step for downstream Chinese tasks such as POS-tagging [2], parsing [3] and information extraction [4]. Chinese word segmentation models are far from being perfect. Although word segmentation can be useful for domains where there are rich training data, the accuracy can be significantly lower for social media, medicine, literature and other domains [5]–[10], causing error propagation to downstream tasks. With the rise of deep learning, neural sequence encoders allow better sentence-level features to be extracted from character sequences alone. As a result, many end-to-end neural Chinese NLP models avoid using Chinese word segmentation [11]. This results in a dilemma between the use of word segmentation and the choice of performing Chinese NLP based on a fully character-based system.

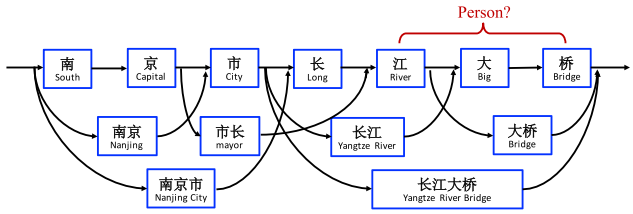


Fig. 1. Word character lattice.

Intuitively, character sequences contain exponential segmentation ambiguities, which can make it challenging to encode precisely useful word information through a neural representation. To address this issue, word information can be a useful addition. Consider for the example the task of named entity recognition. As shown in Figure 1, given the sentence "南京市长江大桥 (Nanjing Yangtze River Bridge)," word sequences such as "长江大桥 (Yangtze River Bridge)," "长江 (Yangtze River)" and "大桥 (Bridge)" can be used to disambiguate potential relevant named entities in a context, such as the person name "江大桥 (Daqiao Jiang)". Although there are ambiguities among potential words such as "市长 (mayor)" and "长江 (Yangtze river)," knowing these candidate words can largely reduce ambiguities embedded in the character sequence alone.

Given the above observations, we consider a neural sentence representation model that enables us to leverage word information without word segmentation. In particular, we assume that a word lexicon is available, which provides possible words given a sentence. Such a lexicon can be obtained from lexical resources such as HowNet [12], knowledge resources such as Baidu Baike[1], and large post-processed corpora with automatic segmentation. Given such a lexicon, possible words in a sentence can be found by matching all sub-sequences to the lexicon. The potential words and the original characters form a lattice structure, which contain both the correct segmentation path and those with incorrect words such as the word "市长 (mayor)" in Figure 2. There are an exponential number of word-character paths in a lattice, with only one correct path.

Manuscript received November 8, 2019; revised March 24, 2020; accepted April 26, 2020. Date of publication April 30, 2020; date of current version June 1, 2020. This work was supported by the National Natural Science Foundation of China under Grant 61976180. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Taro Watanabe. *(Yue Zhang, Yile Wang, and Jie Yang contributed equally to this work.) (Corresponding author: Yue Zhang.)*

Yue Zhang and Yile Wang are with the Zhejiang University, Hangzhou 310027, China, with the School of Engineering, Westlake University, Hangzhou 310024, China, and also with the Westlake Institute for Advanced Study, Hangzhou 310024, China (e-mail: yue.zhang@wias.org.cn; wangyile@westlake.edu.cn).

Jie Yang is with the Harvard Medical School, Harvard University, Cambridge, MA 02138 USA (e-mail: jieynlp@gmail.com).

Digital Object Identifier 10.1109/TASLP.2020.2991544

[1][Online]. Available: https://baike.baidu.com/

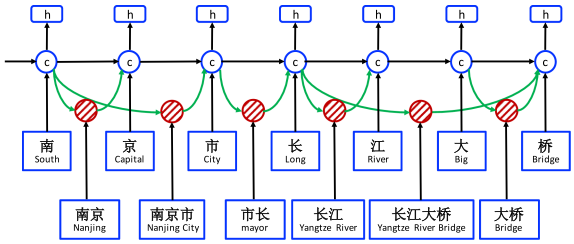


Fig. 2. Lattice LSTM structure.

We leverage a lattice LSTM structure for automatically controlling information flow from the beginning of the sentence to the end. As shown in Figure 2, gated cells are used to dynamically route information from different paths to each character, so that words that are relevant to the training task can be picked up automatically in the representation process for better performance. Compared with character-based and word-based methods, our model has the advantage of using explicit word information over character sequence labeling without suffering from segmentation error. We compare the word-character lattice LSTM model with its sequence LSTM baseline on NER and language modeling tasks.

Inspired by the recent success in pre-trained language models as contextualized word representations [13]–[15], we further consider using lattice LSTM to replace standard LSTM for pre-training a bidirectional LSTM Chinese language model, which serves as a word-informed alternative to character-based ELMo. To this end, two factors must be considered on top of the naive lattice LSTM structure for NER. First, batch training should be enabled for handling large scale training. This is non-trivial since the lattice structure for each sentence can differ. Second, multi-layer support should be considered to allow better encoding power. We tackled these two issues using a novel filled lattice structure, resulting in a MUlti-layer LAttice Network (MULAN) for Chinese representation.

Rich experiments show that our model significantly outperforms both character sequence labeling models and word sequence labeling models using LSTM-CRF, giving the best results over a variety of Chinese NER datasets across different domains. In addition, word-character lattice also gives better results on Chinese language modeling compared with character-based sequence LSTM. Finally, MULAN contextualized embeddings are useful for improving a number of other Chinese tasks, including POS-tagging and syntactic chunking. Significantly better results are obtained as compared to using ELMo contextualized character embeddings on these tasks. We obtain the best-reported results on all relevant benchmarks. Our code and data are released.[2]

This article is a largely extended version of a conference version [16], which presents the basic model for lattice LSTM Chinese NER. In this journal version, we further consider its multi-layer and batch extension, the empirical results on language model training, the pre-training of MULAN contextualized Chinese embeddings, and the empirical study of using MULAN vectors for Chinese NER, POS-tagging, chunking and dependency parsing.

[2][Online]. Available: https://github.com/jiesutd/LatticeLSTM and https://github.com/ylwangy/Lattice4LM

II. Related Work

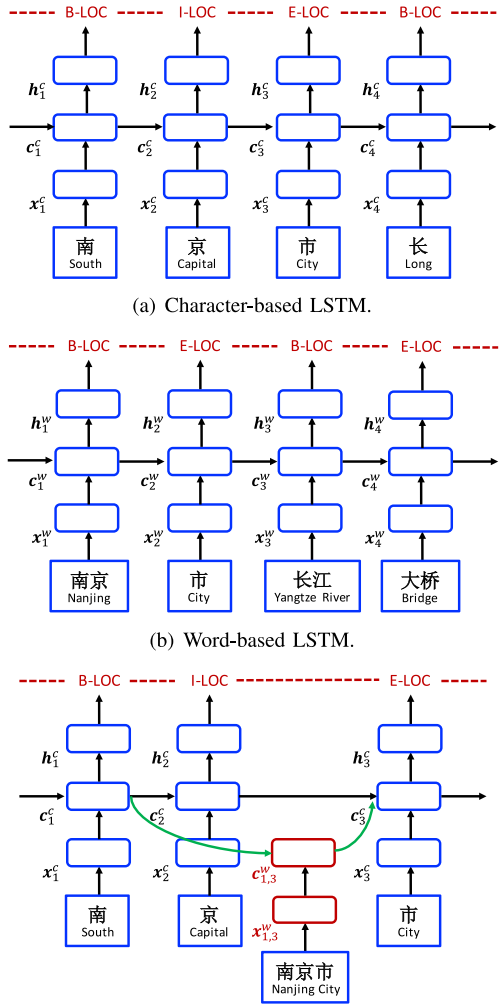
Lattice LSTM: Lattice structured RNNs can be viewed as a natural extension of tree-structured RNNs [17] to DAGs [18], [19]. They have been used to model motion dynamics [20], dependency-discourse DAGs [21], as well as speech tokenization lattice [22] and multi-granularity segmentation outputs [23] for NMT encoders. Compared with existing work, our lattice LSTM is different in both motivation and structure. For example, being designed for character-centric lattice-LSTM-CRF sequence labeling, it has recurrent cells but not hidden vectors for words. To our knowledge, we are the first to design a novel lattice LSTM representation for mixed characters and lexicon words.

NER: [24] attempted to solve the problem using a unidirectional LSTM, which was among the first neural models for NER. [25] used a CNN-CRF structure, obtaining competitive results to the best statistical models. [26] used character CNN to augment a CNN-CRF model. Most recent work leverages an LSTM-CRF architecture. [27] uses hand-crafted spelling features; [28] and [29] use a character CNN to represent spelling characteristics; [30] use a character LSTM instead. [31] build a unified framework NCRF++ which includes all the models above.

Character sequence labeling has been the dominant approach for Chinese NER [32]–[34]. There have been explicit discussions comparing statistical word-based and character-based methods for the task, showing that the latter is empirically a superior choice [35]–[37]. We find that with proper representation settings, the same conclusion holds for neural NER. On the other hand, lattice LSTM is a better choice compared with both word LSTM and character LSTM.

Chinese Language Modeling: Language modeling is a basic task for NLP and it is more challenge for Chinese because of the issue of word segmentation. Zhao *et al.* [38] improved the performance of Chinese language modeling through lexicon generation and lexicon pruning. Buckman *et al.* [39] propose a language modeling paradigm over multi-character chunks, marginalizing over all segmentations of a sentence. Our method is different in using a lattice structure to integrating both character and word information.

Contextualized Embeddings: Neural language modeling has been shown capable of capturing deep semantic features from large unlabeled resources, which can be useful semi-supervised signals for NLP tasks [40], [41]. ELMo [13] provides contextualized word representations generated from word-level language modeling. GPT [14] improves language understanding by generative pre-training based on Transformer [42]. BERT [15] investigates a self-attention-network for pre-training deep bidirectional representations. Our work is in line, building language-model-based contextualized representation pre-trained from large raw text corpus for Chinese. Different from the above methods,



(a) Character-based LSTM.

(b) Word-based LSTM.

(c) Word-Character Lattice LSTM.

Fig. 3. Word/character LSTM and word-character lattice LSTM Models.

however, we integrate both character and word information, generating better contextualized representation without suffering from potential word segmentation errors.

III. Word-Character Lattice LSTM

We take the standard sequential LSTM as our baseline, discussing the representation of a sentence over a character sequence and a word sequence, respectively, where the latter corresponds to a representation using word segmentation. Our word-character lattice LSTM can be viewed as a natural extension to the sequential LSTM structure. This this section, we discuss the three models in a unified notation. Figure 4 gives a structural illustration.

A. Character-Based LSTM

The character-based model is shown in Figure 3(a). It uses an LSTM-CRF model on the character sequence $c_1, c_2, \ldots, c_m$. Each character c_j is represented using

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

$\mathbf{e}^c$ denotes a character embedding lookup table.

A bidirectional LSTM (same structurally as Eq. 11) is applied to $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m$ to obtain $\overrightarrow{\mathbf{h}}_1^c, \overrightarrow{\mathbf{h}}_2^c, \ldots, \overrightarrow{\mathbf{h}}_m^c$ and $\overleftarrow{\mathbf{h}}_1^c, \overleftarrow{\mathbf{h}}_2^c, \ldots, \overleftarrow{\mathbf{h}}_m^c$ in the left-to-right and right-to-left directions, respectively, with two distinct sets of parameters. The hidden vector representation of each character is:

$$\mathbf{h}_j^c = [\overrightarrow{\mathbf{h}}_j^c; \overleftarrow{\mathbf{h}}_j^c] \tag{2}$$

Char + bichar: Character bigrams have been shown useful for representing characters in word segmentation [43], [44]. We augment the character-based model with bigram information by concatenating bigram embeddings with character embeddings:

$$\mathbf{x}_j^c = [\mathbf{e}^c(c_j); \mathbf{e}^b(c_j, c_{j+1})], \tag{3}$$

where $\mathbf{e}^b$ denotes a charater bigram lookup table.

Char + softword: It has been shown that using segmentation as soft features for character-based models can lead to improved performance [4], [45]. We augment the character representation with segmentation information by concatenating segmentation label embeddings to character embeddings:

$$\mathbf{x}_j^c = [\mathbf{e}^c(c_j); \mathbf{e}^s(seg(c_j))], \tag{4}$$

where $\mathbf{e}^s$ represents a segmentation label embedding lookup table. $seg(c_j)$ denotes the segmentation label on the character c_j given by a word segmentor. We use the BMES scheme for representing segmentation [1].

B. Word-Based LSTM

The word-based model is shown in Figure 3(b). It takes the word embedding $\mathbf{e}^w(w_i)$ for representation each word w_i:

$$\mathbf{x}_i^w = \mathbf{e}^w(w_i), \tag{5}$$

where $\mathbf{e}^w$ denotes a word embedding lookup table. A bi-directional LSTM (Eq. 11) is used to obtain a left-to-right sequence of hidden states $\overrightarrow{\mathbf{h}}_1^w, \overrightarrow{\mathbf{h}}_2^w, \ldots, \overrightarrow{\mathbf{h}}_n^w$ and a right-to-left sequence of hidden states $\overleftarrow{\mathbf{h}}_1^w, \overleftarrow{\mathbf{h}}_2^w, \ldots, \overleftarrow{\mathbf{h}}_n^w$ for the words $w_1, w_2, \ldots, w_n$, respectively. Finally, for each word w_i, $\overrightarrow{\mathbf{h}}_i^w$ and $\overleftarrow{\mathbf{h}}_i^w$ are concatenated as its representation:

$$\mathbf{h}_i^w = [\overrightarrow{\mathbf{h}}_i^w; \overleftarrow{\mathbf{h}}_i^w] \tag{6}$$

Integrating character representations: Character CNN [28] and LSTM [30] are used for representing the character sequence within a word as two alternatives. Denoting the representation of characters within w_i as $\mathbf{x}_i^c$, a new word representation is obtained by concatenation of $\mathbf{e}^w(w_i)$ and $\mathbf{x}_i^c$:

$$\mathbf{x}_i^w = [\mathbf{e}^w(w_i); \mathbf{x}_i^c] \tag{7}$$

Word + char LSTM: Denoting the embedding of each input character as $\mathbf{e}^c(c_j)$, we use a bi-directional LSTM (Eq. 11) to learn hidden states $\overrightarrow{\mathbf{h}}_{t(i,1)}^c, \ldots, \overrightarrow{\mathbf{h}}_{t(i,len(i))}^c$ and $\overleftarrow{\mathbf{h}}_{t(i,1)}^c, \ldots, \overleftarrow{\mathbf{h}}_{t(i,len(i))}^c$ for the characters $c_{t(i,1)}, \ldots, c_{t(i,len(i))}$ of w_i, where $len(i)$ denotes the number of characters in w_i. The final character representation for w_i is:

$$\mathbf{x}_i^c = [\overrightarrow{\mathbf{h}}_{t(i,len(i))}^c; \overleftarrow{\mathbf{h}}_{t(i,1)}^c] \tag{8}$$

Word + char LSTM′: We investigate a variation of word + char LSTM model that uses a **single** LSTM to obtain $\overrightarrow{\mathbf{h}}_j^c$ and $\overleftarrow{\mathbf{h}}_j^c$ for each c_j. It is similar with the structure of [46] but not uses the highway layer. The same LSTM structure as defined in Eq. 11 is used, and the same method as Eq. 8 is used to integrate character hidden states into word representations.

Word + char CNN: A standard CNN [47] structure is used on the character sequence of each word to obtain its character representation $\mathbf{x}_i^c$. Denoting the embedding of character c_j as $\mathbf{e}^c(c_j)$, the vector $\mathbf{x}_i^c$ is given by:

$$\mathbf{x}_i^c = \max_{t(i,1)\leq j\leq t(i,len(i))} (\mathbf{W}_{\text{CNN}}^\top \begin{bmatrix} \mathbf{e^c}(\mathbf{c_{j-\frac{ke-1}{2}}}) \\ \cdots \\ \mathbf{e}^c(c_{j+\frac{ke-1}{2}}) \end{bmatrix} + \mathbf{b}_{\text{CNN}}), \quad (9)$$

where $\mathbf{W}_{\text{CNN}}$ and $\mathbf{b}_{\text{CNN}}$ are parameters, $ke = 3$ is the kernel size and max denotes max pooling.

C. Word-Character Lattice LSTM

The overall structure of the word-character lattice model is shown in Figure 2, which can be viewed as an extension of the character-based model, integrating word-based cells and additional gates for controlling information flow.

Shown in Figure 3(c), the input to the model is a character sequence $c_1, c_2, \ldots, c_m$, together with all character subsequences that match words in a lexicon $\mathbb{D}$. The lexicon $\mathbb{D}$ can be built by collecting all the words from the automatically segmented large raw text. Using $w_{b,e}^d$ to denote such a subsequence that begins with character index b and ends with character index e, the segment $w_{1,2}^d$ in Figure 1 is "南京 (Nanjing)" and $w_{7,8}^d$ is "大桥 (Bridge)".

Four types of vectors are involved in the model, namely *input vectors*, *output hidden vectors*, *cell vectors* and *gate vectors*. As basic components, a character input vector is used to represent each character c_j as in the character-based model:

$$\mathbf{x}_j^c = \mathbf{e}^c(c_j) \quad (10)$$

The basic recurrent structure of the model is constructed using a character cell vector $\mathbf{c}_j^c$ and a hidden vector $\mathbf{h}_j^c$ on each c_j, where $\mathbf{c}_j^c$ serves to record recurrent information flow from the beginning of the sentence to c_j and $\mathbf{h}_j^c$ is used for sequence labelling representation.

The basic recurrent LSTM functions are:

$$\begin{bmatrix} \mathbf{i_j^c} \\ \mathbf{o}_j^c \\ \mathbf{f}_j^c \\ \widetilde{\mathbf{c}}_j^c \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{bmatrix} \left(\mathbf{W}^{c\top} \begin{bmatrix} \mathbf{x_j^c} \\ \mathbf{h}_{j-1}^c \end{bmatrix} + \mathbf{b}^c \right)$$
$$\mathbf{c}_j^c = \mathbf{f}_j^c \odot \mathbf{c}_{j-1}^c + \mathbf{i}_j^c \odot \widetilde{\mathbf{c}}_j^c \quad (11)$$
$$\mathbf{h}_j^c = \mathbf{o}_j^c \odot tanh(\mathbf{c}_j^c)$$

where $\mathbf{i}_j^c$, $\mathbf{f}_j^c$ and $\mathbf{o}_j^c$ denote a set of input, forget and output gates, respectively. $\mathbf{W}^{c\top}$ and $\mathbf{b}^c$ are model parameters. $\sigma()$ represents the sigmoid function.

Different from the character-based model, however, the computation of $\mathbf{c}_j^c$ now considers lexicon subsequences $w_{b,e}^d$ in the sentence. In particular, each subsequence $w_{b,e}^d$ is represented using

$$\mathbf{x}_{b,e}^w = \mathbf{e}^w(w_{b,e}^d), \quad (12)$$

where $\mathbf{e}^w$ denotes the same word embedding lookup table as in Section III-B.

In addition, a word cell $\mathbf{c}_{b,e}^w$ is used to represent the recurrent state of $\mathbf{x}_{b,e}^w$ from the beginning of the sentence. The value of $\mathbf{c}_{b,e}^w$ is calculated by:

$$\begin{bmatrix} \mathbf{i_{b,e}^w} \\ \mathbf{f}_{b,e}^w \\ \widetilde{\boldsymbol{c}}_{b,e}^w \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ tanh \end{bmatrix} \left(\mathbf{W}^{w\top} \begin{bmatrix} \mathbf{x_{b,e}^w} \\ \mathbf{h}_b^c \end{bmatrix} + \mathbf{b}^w \right)$$
$$\mathbf{c}_{b,e}^w = \mathbf{f}_{b,e}^w \odot \mathbf{c}_b^c + \mathbf{i}_{b,e}^w \odot \widetilde{\boldsymbol{c}}_{b,e}^w \quad (13)$$

where $\mathbf{i}_{b,e}^w$ and $\mathbf{f}_{b,e}^w$ are a set of input and forget gates. There is no output gate for word cells since labeling is performed only at the character level.

With $\mathbf{c}_{b,e}^w$, there are more recurrent paths for information flow into each $\mathbf{c}_j^c$. For example, in Figure 2, input sources for $\mathbf{c}_7^c$ include $\mathbf{x}_7^c$ (桥 Bridge), $\mathbf{c}_{6,7}^w$ (大桥 Bridge) and $\mathbf{c}_{4,7}^w$ (长江大桥 Yangtze River Bridge).[3] We link all $\mathbf{c}_{b,e}^w$ with $b \in \{b' | w_{b',e}^d \in \mathbb{D}\}$ to the cell $\mathbf{c}_e^c$. We use an additional gate $\mathbf{i}_{b,e}^c$ for each subsequence cell $\mathbf{c}_{b,e}^w$ for controlling its contribution into $\mathbf{c}_{b,e}^c$:

$$\mathbf{i}_{b,e}^c = \sigma \left(\mathbf{W}^{l\top} \begin{bmatrix} \mathbf{x_e^c} \\ \mathbf{c}_{b,e}^w \end{bmatrix} + \mathbf{b}^l \right) \quad (14)$$

The calculation of cell values $\mathbf{c}_j^c$ thus becomes

$$\mathbf{c}_j^c = \sum_{b\in\{b'|w_{b',j}^d\in\mathbb{D}\}} \boldsymbol{\alpha}_{b,j}^c \odot \boldsymbol{c}_{b,j}^w + \boldsymbol{\alpha}_j^c \odot \widetilde{\boldsymbol{c}}_j^c \quad (15)$$

In Eq. 15, the gate values $\mathbf{i}_{b,j}^c$ and $\mathbf{i}_j^c$ are normalised to $\boldsymbol{\alpha}_{b,j}^c$ and $\boldsymbol{\alpha}_j^c$ by setting the sum to $\mathbf{1}$.

$$\boldsymbol{\alpha}_{b,j}^c = \frac{exp(\mathbf{i}_{b,j}^c)}{exp(\mathbf{i}_j^c) + \sum_{b'\in\{b''|w_{b'',j}^d\in\mathbb{D}\}} exp(\mathbf{i}_{b',j}^c)}$$
$$\boldsymbol{\alpha}_j^c = \frac{exp(\mathbf{i}_j^c)}{exp(\mathbf{i}_j^c) + \sum_{b'\in\{b''|w_{b'',j}^d\in\mathbb{D}\}} exp(\mathbf{i}_{b',j}^c)} \quad (16)$$

IV. NER and Language Modeling

We perform named entity recognition (NER) and language modeling as two tasks to compare the effectiveness of word-character lattice LSTM with sequential LSTM. In particular, for NER, a CRF layer is used on top of the LSTM representations for predicting the output. For language modeling we use a simple softmax layer for predicting the next character given a sequence of characters.

[3]We experimented with alternative configurations on indexing word and character path links, finding that this configuration gives the best results in preliminary experiments. Single-character words are excluded; the final performance drops slightly after integrating single-character words.

A. NER

The best NER models [27], [28], [30] use LSTM-CRF as the main network structure. We compare lattice LSTM with its sequential counterparts under this framework. Formally, denote an input sentence as $s = c_1, c_2, \ldots, c_m$, where c_j denotes the jth character. s can further be seen as a word sequence $s = w_1, w_2, \ldots, w_n$, where w_i denotes the ith word in the sentence, obtained using a Chinese segmentor. We use $t(i, k)$ to denote the index j for the kth character in the ith word in the sentence. Take the sentence in Figure 1 for example. If the segmentation is "南京市 长江大桥," and indices are from 1, then $t(2, 1) = 4$ (长) and $t(1, 3) = 3$ (市). We use the BIOES tagging scheme [48] for both word-based and character-based NER tagging.[4]

Given the representation methods in Section III, a standard CRF layer is used on top of $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_\tau$, where τ is n for character-based and lattice-based models and m for word-based models. The probability of a label sequence $y = l_1, l_2, \ldots, l_\tau$ is

$$P(y|s) = \frac{exp(\sum_i (\mathbf{W}^{l_i}_{\text{CRF}} \mathbf{h}_i + b^{(l_{i-1}, l_i)}_{\text{CRF}}))}{\sum_{y'} exp(\sum_i (\mathbf{W}^{l'_i}_{\text{CRF}} \mathbf{h}_i + b^{(l'_{i-1}, l'_i)}_{\text{CRF}}))} \tag{17}$$

where y' represents an arbitrary label sequence, and $\mathbf{W}^{l_i}_{\text{CRF}}$ is a model parameter specific to l_i, and $b^{(l_{i-1}, l_i)}_{\text{CRF}}$ is a bias specific to l_{i-1} and l_i.

We use the first-order Viterbi algorithm to find the highest scored label sequence over a word-based or character-based input sequence. Given a set of manually labeled training data $\{(s_i, y_i)\}|_{i=1}^N$, sentence-level log-likelihood loss with L_2 regularization is used to train the model:

$$L = \sum_{i=1}^{N} log(P(y_i|s_i)) + \frac{\lambda}{2}||\Theta||^2, \tag{18}$$

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

During back-propagation training, we compute local gradients on $\mathbf{h}_i$ over the CRF layer, and then train the sequential and lattice LSTM representation layers by further back-propagated gradients.

B. Language Modeling

Given a partial sentence $s = c_1, \ldots, c_{i-1}$, our goal is to predict c_i. To this end, both a baseline LSTM and a character-word lattice LSTM in Section III can be used to encode s, obtaining a sequence of hidden states $\mathbf{h}^c_1, \ldots, \mathbf{h}^c_{i-1}$. In this process, only lexicon words that match the partial sentence s are used to build the lattice structure over s; we do not consider partial matches, where a sub word from the beginning of a lexicon word matches the last few characters in s.

Similar to a standard character-based LSTM language model, the last hidden state $\mathbf{h}^c_{i-1}$ is used to predict c_i

$$P(c_i|s) = \text{softmax}\left(\mathbf{W}^o \mathbf{h}^c_{i-1}\right)$$

where $\mathbf{W}^o$ is a model parameter.

[4]To keep the figure concise, we (i) do not show gate cells, which uses $\mathbf{h}_{t-1}$ for calculating $\mathbf{c}_t$; (ii) only show one direction.

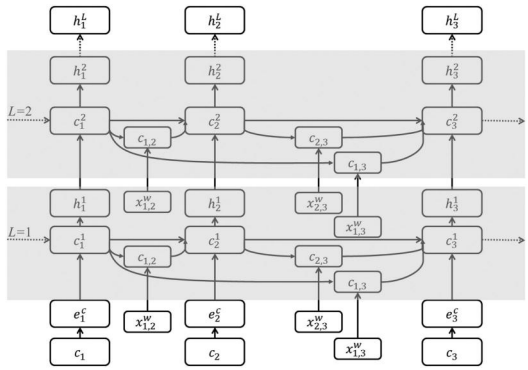


Fig. 4. The structure of multi-layer stacked lattice LSTM.

Given a set of training data $D = \{s_i\}|_{i=1}^N$, we maximize the log-likelihood of the data

$$J = \sum_{i=1}^{N} \sum_{j=1}^{|s_i|} P(c^i_j | c^i_1, \ldots, c^i_{j-1})$$

where $s_i = c^i_1, \ldots, c^i_{|s_i|}$.

Similar to the case of NER, during training local gradients are first back-propagated to $\mathbf{h}_i$ from the softmax layer, and then further back-propagated for training the sequential and lattice LSTM structures in Section III.

V. Multi-Layer Lattice Language Model and Contextualized Chinese Representation

Language modeling offers a more general testbed for representation learning, which serves as a basis for training contextualized embeddings [13]. Compared with training for small-scale tasks, training for contextualized embedding models yet poses two main challenges. First, the model should accommodate a significantly larger amount of training data, and therefore multiple encoding layers can be necessary for better fitting. Second, batch training can be necessary to allow faster convergence. This section discusses multi-layer extension and batch training, respectively, before language model training and contextualized representation.

A. Multi-Layer Extension

We build multi-layer lattice LSTM by feeding the hidden states $\mathbf{h}^c_j$ and subsequence representations $\mathbf{x}^w_{b,e}$ into a new layer of lattice LSTM as input to generate a higher level meaning of characters. Multiple layers of lattice LSTM can be stacked, resulting in a hierarchy of hidden states $\mathbf{h}^{c,1}_j$, $\mathbf{h}^{c,2}_j$, ..., $\mathbf{h}^{c,l}_j$ for each character c_j, where $\mathbf{h}^{c,l}_j$ denotes the hidden state $\mathbf{h}^c_j$ of layer number l. One example multi-layer lattice LSTM model is shown in Figure 4.

B. Batch Training

Because different sentences have different lattice structures, cell computation cannot be directly batched. For example, as Figure 5 shows, the input lattice structures for the first sentence are "南京 (Nanjing)" and "南京市 (Nanjing City)," while for the

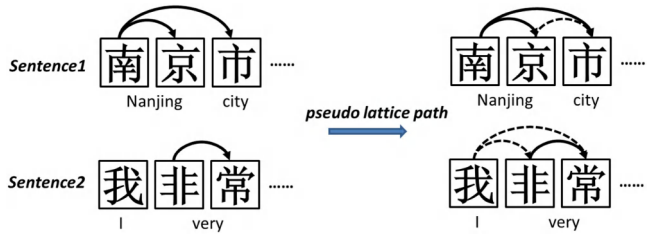


Fig. 5. Illustration of pseudo lattice paths.

second sentence the result is "非常 (very)," which have different number, position and length. To solve this problem, we extend all lattice structures into a filled lattice structure, by adding pseudo lattice paths over character spans that do not form words. To reduce runtime complexity, we set of limit to the maximum word size in the lattice. The resulting lattice structure is then uniform across sentences, where all paths between a character window are filled with word or pseudo word links.

Formally, the model structure is modified by adding the paths between all c_i and c_j ($i \in \{j\text{-}len, j\text{-}len{+}1, ..., j\text{-}1\}$) within a window of size len no matter whether subsequence $w_{b,e}^d$ are legal words or not, where len is a pre-defined maximum Chinese word length. To make the lattice structure differentiable, we use the $\mathbf{0}$ vector for representing the none words, thus the general representation of all word subsequence $w_{b,e}^d$ are as follows:

$$\mathbf{x}_{b,e}^{w} = \begin{cases} \mathbf{e^w}(\mathbf{w_{b,e}^d}), & \text{if} w_{b,e}^d \in \mathbf{e}^w; \\ \mathbf{0}, & \text{if} w_{b,e}^d \notin \mathbf{e}^w. \end{cases} \tag{19}$$

The hidden vectors $\mathbf{h}_j^c$ are still computed as described by Eq. 11. During training, loss values back-propagate to the parameters $\mathbf{W}^c, \mathbf{b}^c, \mathbf{W}^w, \mathbf{b}^w, \mathbf{W}^l$ and $\mathbf{b}^l$ allowing the model to dynamically focus on more relevant words during labelling.

C. Contextualized Representation

Large-scaled pretrained multi-layer bi-directional LSTM language models have been used for contextualized representations of words for English [13]. Such representation can also be pretrained over characters, resulting in contextualized Chinese embeddings based on character language models. Our multi-layer lattice LSTM language model can be used to replace a multi-layer LSTM language model to this goal, and we call the representation MUlti-layer LAttice Network (MULAN) contextualized representation.

Following [13], given a sentence $s = c_1, \ldots, c_m$, the contextualized representation of c_j is generated by using a weighted combination of hidden states from different layers:

$$\mathbf{e}_j^{LM} = \sum_{k=0}^{l} \beta_k^{task} \mathbf{h_j^{c,k}} \tag{20}$$

where $\mathbf{h}_0$ stands for the embedding layer, β_k^{task} are the task related softmax-normalized weights. k is the layer index.

Our representation $\mathbf{e}_j^{LM}$ is character-based, however, it encodes both character and word embedding information. The contextualized representation can be directly used as external input to NLP models, enhancing the power of input embedding layer representation or the output layer representation. In particular, in the former case, MULAN representations (Eq. 20) for each word is concatenated with the input embeddings of each word as new input embeddings for a task; in the latter case, we concatenate MULAN embeddings for each word to the last hidden layers of the end-task encoder, instead of the input layer. We find empirically that enriching the input representation works better.

TABLE I
STATISTICS OF DATASETS

Dataset	Type	Train	Dev	Test
OntoNotes	Sentence	15.7k	4.3k	4.3k
	Char	491.9k	200.5k	208.1k
MSRA	Sentence	46.4k	–	4.4k
	Char	2169.9k	–	172.6
Weibo	Sentence	1.4k	0.27k	0.27k
	Char	73.8k	14.5k	14.8k
resume	Sentence	3.8k	0.46k	0.48k
	Char	124.1k	13.9k	15.1k

VI. EXPERIMENTS ON NER

We carry out an extensive set of experiments to investigate the effectiveness of word-character lattice LSTMs for Chinese NER across different domains. In addition, we aim to empirically compare word-based and character-based neural Chinese NER under different settings. Standard precision (P), recall (R) and F1-score (F1) are used as evaluation metrics.

A. Experimental Settings

Data: Four datasets are used in this paper, which include OntoNotes 4 [49], MSRA [50] Weibo NER [51], [52] and a Chinese resume dataset that we annotate. Statistics of the datasets are shown in Table I. We take the same data split as [53] on OntoNotes. The development set of OntoNotes is used for reporting development experiments. While the OntoNotes and MSRA datasets are in the news domain, the Weibo NER dataset is drawn from the social media website Sina Weibo.[5]

For more variety in test domains, we collected a resume dataset from Sina Finance,[6] which consists of resumes of senior executives from listed companies in the Chinese stock market. We randomly selected 1027 resume summaries and manually annotated 8 types of named entities with YEDDA system [54]. Statistics of the dataset is shown in Table II. The inter-annotator agreement is 97.1%. We release this dataset as a resource for further research.

Segmentation: For the OntoNotes and MSRA datasets, gold-standard segmentation is available in the training sections. For OntoNotes, gold segmentation is also available for the development and test sections. On the other hand, no segmentation is available for the MSRA test sections, nor the Weibo / resume datasets. As a result, OntoNotes is leveraged for studying oracle situations where gold segmentation is given. We use the neural

[5][Online]. Available: https://www.weibo.com/
[6][Online]. Available: http://finance.sina.com.cn/stock/index.shtml

TABLE II
DETAILED STATISTICS OF RESUME NER

Statistics	Train	Dev	Test
Country	260	33	28
Educational Institution	858	106	112
Location	47	2	6
Personal Name	952	110	112
Organization	4611	523	553
Profession	287	18	33
Ethnicity Background	115	15	14
Job Title	6308	690	772
Total Entity	13438	1497	1630

TABLE III
HYPER-PARAMETER VALUES FOR NER

Parameter	Value	Parameter	Value
char emb size	50	bigram emb size	50
lattice emb size	200	LSTM hidden	200
char dropout	0.5	lattice dropout	0.5
LSTM layer	1	regularization λ	1e-8
learning rate lr	0.015	lr decay	0.05

word segmentor of [44] to automatically segment the development and test sets for word-based NER. In particular, for the OntoNotes and MSRA datasets, we train the segmentor using gold segmentation on their respective training sets. For Weibo and resume, we take the best model of [44] off the shelf[7], which is trained using CTB 6.0 [55].

Word Embeddings: We pretrain word embeddings using word2vec [56] over automatically segmented Chinese Giga-Word[8], obtaining 704.4 k words in a final lexicon. In particular, the number of single-character, two-character and three-character words are 5.7 k, 291.5 k, 278.1 k, respectively. The embedding lexicon is released alongside our code and models as a resource for further research. Word embeddings are fine-tuned during NER training. Character and character bigram embeddings are pretrained on Chinese Giga-Word using word2vec and fine-tuned at model training.

Hyper-Parameter Settings: Table III shows the values of hyper-parameters for our models, which as fixed according to previous work in the literature without grid-search adjustments for each individual dataset. In particular, the embedding sizes are set to 50 and the hidden size of LSTM models to 200. Dropout [57] is applied to both word and character embeddings with a rate of 0.5. Stochastic gradient descent (SGD) is used for optimization, with an initial learning rate of 0.015 and a decay rate of 0.05.

B. Development Experiments

We compare various model configurations on the OntoNotes development set, in order to select the best settings for word-based and character-based NER models, and to learn the influence of lattice word information on character-based models.

Character-based NER: As shown in Table IV, without using word segmentation, a character-based LSTM-CRF model gives a development F1-score of 62.47%. Adding character-bigram and softword representations as described in Section III-A increases the F1-score to 67.63% and 65.71%, respectively, demonstrating the usefulness of both sources of information. In addition, a combination of both gives a 69.64% F1-score, which is the best among various character representations. We thus choose this model in the remaining experiments.

[7][Online]. Available: https://github.com/jiesutd/RichWordSegmentor
[8][Online]. Available: https://catalog.ldc.upenn.edu/LDC2011T13

TABLE IV
DEVELOPMENT RESULTS FOR NER

Input	Models	P	R	F1
Auto seg	Word baseline	73.20	57.05	64.12
	+char LSTM	71.98	65.41	68.54
	+char LSTM′	71.08	65.83	68.35
	+char+bichar LSTM	72.63	67.60	70.03
	+char CNN	73.06	66.29	69.51
	+char+bichar CNN	72.01	65.50	68.60
No seg	Char baseline	67.12	58.42	62.47
	+softword	69.30	62.47	65.71
	+bichar	71.67	64.02	67.63
	+bichar+softword	72.64	66.89	69.64
	Lattice	**74.64**	**68.83**	**71.62**

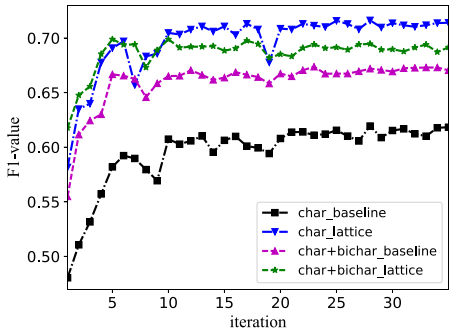


Fig. 6. F1 against the training iteration number.

Word-Based NER: Table IV shows a variety of different settings for word-based Chinese NER. With automatic segmentation, a word-based LSTM CRF baseline gives a 64.12% F1-score, which is higher compared to the character-based baseline. This demonstrates that both word information and character information are useful for Chinese NER. The two methods of using character LSTM to enrich word representations in Section III-B, namely word+char LSTM and word+char LSTM′, lead to similar improvements.

A CNN representation of character sequences gives a slightly higher F1-score compared to LSTM character representations. On the other hand, further using character bigram information leads to increased F1-score over word+char LSTM, but decreased F1-score over word+char CNN. A possible reason is that CNN inherently captures character n-gram information. As a result, we use word+char+bichar LSTM for word-based NER in the remaining experiments, which gives the best development results, and is structurally consistent with the state-of-the-art English NER models in the literature.

Word-Character Lattice NER: Figure 6 shows the F1-score of character-based and lattice-based models against the number of training iterations. We include models that use concatenated character and character bigram embeddings, where bigrams can play a role in disambiguating characters. As can be seen from the figure, lattice word information is useful for improving

TABLE V
MAIN RESULTS ON ONTONOTES

Input	Models	P	R	F1
Gold seg	Yang et al.[60]	65.59	71.84	68.57
	Yang et al.[60]*†	72.98	**80.15**	**76.40**
	Che et al.[54]*	77.71	72.51	75.02
	Wang et al.[59]*	76.43	72.32	74.32
	Word baseline	76.66	63.60	69.52
	+char+bichar LSTM	**78.62**	73.13	75.77
Auto seg	Word baseline	72.84	59.72	65.63
	+char+bichar LSTM	73.36	70.12	71.70
No seg	Char baseline	68.79	60.35	64.30
	+bichar+softword	74.36	69.43	71.81
	Lattice	**76.35**	**71.56**	**73.88**

character-based NER, improving the best development result from 62.5% to 71.6%. On the other hand, the bigram-enhanced lattice model does not lead to further improvements compared with the original lattice model. This is likely because words are better sources of information for character disambiguation compared with bigrams, which are also ambiguous.

As shown in Table IV, the lattice LSTM-CRF model gives a development F1-score of 71.62%, which is significantly[9] higher compared with both the word-based and character-based methods, despite that it does not use character bigrams or word segmentation information. The fact that it significantly outperforms char+softword shows the advantage of lattice word information as compared with segmentor word information.

C. Final Results

OntoNotes: The OntoNotes test results are shown in Table V.[10] With gold-standard segmentation, our word-based methods give competitive results to the state-of-the-art on the dataset [53], [58], which leverage bilingual data. This demonstrates that LSTM-CRF is a competitive choice for word-based Chinese NER, as it is for other languages. In addition, the results show that our word-based models can serve as highly competitive baselines. With automatic segmentation, the F1-score of word+char+bichar LSTM decreases from 75.77% to 71.70%, showing the influence of segmentation to NER. Consistent with observations on the development set, adding lattice word information leads to an 88.81% $\rightarrow$ 93.18% increasement of F1-score over the character baseline, as compared with 88.81% $\rightarrow$ 91.87% by adding bichar+softword. The lattice model gives significantly the best F1-score on automatic segmentation.

MSRA: Results on the MSRA dataset are shown in Table VI. For this benchmark, no gold-standard segmentation is available on the test set. Our chosen segmentor gives 95.93% accuracy on 5-fold cross-validated training set. The best statistical models on the dataset leverage rich handcrafted features [60]–[62] and character embedding features [33]. [34] exploit neural LSTM-CRF with radical features.

[9]We use a p-value of less than 0.01 from pairwise t-test to indicate statistical significance.

[10]In Table V VI and VII, we use * to denote a model with external labeled data for semi-supervised learning. † means that the model also uses discrete features.

TABLE VI
MAIN RESULTS ON MSRA

Models	P	R	F1
Chen et al.[61]	91.22	81.71	86.20
Zhang et al.[62]*	92.20	90.18	91.18
Zhou et al.[63]	91.86	88.75	90.28
Lu et al.[33]	–	–	87.94
Dong et al.[34]	91.28	90.62	90.95
Word baseline	90.57	83.06	86.65
+char+bichar LSTM	91.05	89.53	90.28
Char baseline	90.74	86.96	88.81
+bichar+softword	92.97	90.80	91.87
Lattice	**93.57**	**92.79**	**93.18**

TABLE VII
WEIBO NER RESULTS

Models	NE	NM	Overall
Peng et al.[52]	51.96	61.05	56.05
Peng et al.[4]*	**55.28**	**62.97**	**58.99**
He et al.[53]	50.60	59.32	54.82
He et al.[64]*	54.50	62.17	58.23
Word baseline	36.02	59.38	47.33
+char+bichar LSTM	43.40	60.30	52.33
Char baseline	46.11	55.29	52.77
+bichar+softword	50.55	60.11	56.75
Lattice	**53.04**	**62.25**	**58.79**

TABLE VIII
MAIN RESULTS ON RESUME NER

Models	P	R	F1
Word baseline	93.72	93.44	93.58
+char+bichar LSTM	94.07	94.42	94.24
Char baseline	93.66	93.31	93.48
+bichar+softword	94.53	**94.29**	94.41
Lattice	**94.81**	94.11	**94.46**

Compared with the existing methods, our word-based and character-based LSTM-CRF models give competitive accuracies. The lattice model significantly outperforms both the best character-based and word-based models ($p < 0.01$), achieving the best result on this standard benchmark.

Weibo/Resume: Results on the Weibo NER dataset are shown in Table VII, where NE, NM and Overall denote F1-scores for named entities, nominal entities (excluding named entities) and both, respectively. Gold-standard segmentation is not available for this dataset. Existing state-of-the-art systems include [4] and [63], who explore rich embedding features, cross-domain and semi-supervised data, some of which are orthogonal to our model.[11]

Results on the resume NER test data are shown in Table VIII. Consistent with observations on OntoNotes and MSRA, the lattice model significantly outperforms both the word-based mode and the character-based model for Weibo and resume ($p < 0.01$), giving state-of-the-art results.

D. Discussion

F1 Against Sentence Length: Figure 7 shows the F1-scores of the baseline models and lattice LSTM-CRF on the OntoNotes dataset. The character-based baseline gives relatively stable F1-scores over different sentence lengths, although the performances are relatively low. The word-based baseline gives substantially higher F1-scores over short sentences, but lower

[11]The results of [4], [51] are taken from [64].

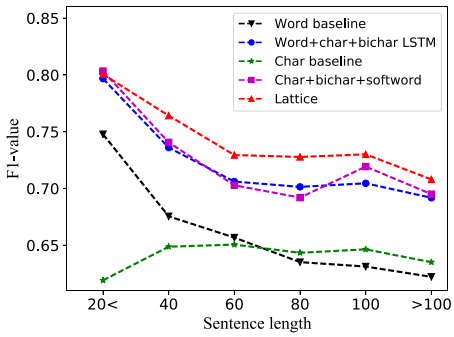


Fig. 7. F1 against sentence length for NER.

TABLE IX
NER EXAMPLES. RED AND GREEN REPRESENT INCORRECT AND CORRECT ENTITIES, RESPECTIVELY

Sentence (truncated)	卸下东莞台协会长职务后 After stepping down as president of Taiwan Association in Dongguan.
Correct Segmentation	卸下 东莞 台 协 会长 职务 后 step down, Dongguan, Taiwan, association, president, role, after
Auto Segmentation	卸下 东莞 台 协会长 职务 后 step down, Dongguan, Taiwan, association president, role, after
Lattice words	卸下 下东 东莞 台协会 协会 会长 长职 职务 step down, incorrect word, Dongguan, Taiwan association, association, president, permanent job, role
Word+char+bichar LSTM	卸下 东莞 $_{GPE}$ 台 $_{GPE}$协会长职务后 … Dongguan $_{GPE}$ Taiwan $_{GPE}$ …
Char+bichar+softword	卸下 东莞台协会 $_{ORG}$长职务后 … Taiwan Association in Dongguan $_{ORG}$ …(ungrammatical)
Lattice	卸下 东莞台协 $_{ORG}$会长职务后 … Taiwan Association in Dongguan $_{ORG}$ …

F1-scores over long sentences, which can be because of lower segmentation accuracies over longer sentences. Both word+char+bichar and char+bichar+softword give better performances compared to their respective baselines, showing that word and character representations are complementary for NER. The accuracy of lattice also decreases as the sentence length increases, which can result from exponentially increasing number of word combinations in the lattice. Compared with word+char+bichar and char+bichar+softword, the lattice model shows more robustness to increased sentence lengths, demonstrating the more effective use of word information.

Case Study: Table IX shows a case study comparing char+bichar+softword, word+char+bichar and the lattice model. In the example, there is much ambiguity around the named entity "东莞台协 (Taiwan Association in Dongguan)". Word+char+bichar yields the entities "东莞 (Dongguan)" and "台 (Taiwan)" given that "东莞台协 (Taiwan Association in Dongguan)" is not in the segmentor output. Char+bichar+softword recognizes "东莞台协会 (Taiwan Association in Dongguan)," which is valid on its own, but leaves the phrase "长职务后 (Long duties after)" ungrammatical. In contrast, the lattice model detects the organization name correctly, thanks to the lattice words "东莞 (Dongguan)," "会长 (President)" and "职务 (role)". There are also irrelevant words such as "台协会 (Taiwan Association)" and "下东 (noisy word)" in the lexicon, which did not affect NER results.

Note that both word+char+bichar and lattice use the same source of word information, namely the same pretrained word embedding lexicon. However, word+char+bichar first uses the lexicon in the segmentor, which imposes hard constrains (i.e. fixed words) to its subsequence use in NER. In contrast, lattice LSTM has the freedom of considering all lexicon words.

TABLE X
ENTITIES IN LEXICON

Dataset	Split	#Entity	#Match	Ratio (%)	ER (%)
OntoNotes	Train	13.4k	9.5k	71.04	–
	Test	7.7k	6.0k	78.72	7.34
MSRA	Train	74.7k	54.3k	72.62	–
	Test	6.2k	4.6k	73.76	16.11
Weibo (all)	Train	1.9k	1.1k	58.83	–
	Test	414	259	62.56	4.72
resume	Train	13.4k	3.8k	28.55	–
	Test	1.6k	483	29.63	0.89

Entities in lexicon: Table X shows the total number of entities and their respective match ratios in the lexicon. The error reductions (ER) of the final lattice model over the best character-based method (i.e. "+bichar+softword") are also shown. It can be seen that error reductions have a correlation between matched entities in the lexicon. In this respect, our automatic lexicon also played to some extent the role of a gazetteer [29], [48], but not fully since there is no explicit knowledge in the lexicon which tokens are entities. The ultimate disambiguation power still lies in the lattice encoder and supervised learning.

The quality of the lexicon may affect the accuracy of our NER model since noise words can potentially confuse NER. On the other hand, our lattice model can potentially learn to select more correct words during NER training. We leave the investigation of such influence to future work.

VII. EXPERIMENTS ON LM AND MULAN EMBEDDINGS

In this section, we report comparisons between lattice LSTM and bi-LSTM on language modeling, and the corresponding contextualized representations, namely MULAN and ELMo, on Chinese sequence labeling tasks including POS tagging, chunking, NER and dependency parsing. Note that we report experiments for both language modeling and experiments on contextualized embeddings in the same section, due to the reason that contextualized embeddings are trained using the model structure of language modeling. As a result, we keep the same model configurations and hyper-parameter settings for the two set of experiments. We also try to keep other experimental settings consistent for the two sets of empirical investigation.

A. Experimental Settings

Language Modeling: We study the performance of character-based, word-based and lattice LSTM models on the development data for Chinese language modeling. We subsample 1 million sentences from the Chinese Gigaword corpus Fifth Edition[12] for language modeling. Sentences with a length of 30 to 40 characters are selected. <*bos*> and <*eos*> tokens are added to the two ends of each sentence. The word segmentation model of Yang *et al.* [44] is applied for word models, and words that appear less than five times are replaced with <*unk*>. The total

[12][Online]. Available: https://catalog.ldc.upenn.edu/LDC2011T13

TABLE XI
HYPER-PARAMETER VALUES FOR LANGUAGE MODELING

Parameter	Value	Parameter	Value
char emb size	300	hidden emb size	200
lattice emb size	200	sequence length	40
dropout rate	0.1	gradient clip	5
LSTM layer	2	regularization λ	0
learning rate lr	1e-4	lr decay	0.2

vocabulary size for characters and words are 7048 and 89675, respectively. Standard per character perplexity (PPL) is used as our evaluation metric for language modeling in both the forward and the backward direction. Word perplexities are converted into character perplexities $cppl$ for direct comparison:

$$cppl = 2^{-\frac{1}{n_i}\sum_{j=1}^{n_i}\frac{1}{1+log_2|w_j|}log_2 P(w_j|w_{j-1},...,w_1)}$$

where $|w_j|$ is the length of word w_j.

We use the generated lexicon vocabulary $\mathbb{D}$ from Song *et al.* [65] for word embedding. Adam [66] is used for optimization. During training, word embeddings are fixed while character embeddings are fine-tuned. Dropout [57] is applied to all input embedding layers, with a rate of 0.1. The hyper-parameter settings of both sequential and lattice LSTM models are shown in Table XI, which are selected on the language modeling development data using the respective methods. It is worth noting that different from the hyper-parameter settings for NER, the number of LSTM layers that gives the best development results for language modeling is 2, which reflects the fact that language modeling is a more challenging task with a larger dataset to fit.

Contextualized embeddings: We train contextualized embeddings over the Gigaword dataset by using both the ELMo model and the MULAN model discussed in Section III. We use the hyper-parameter settings for language modeling directly for training contextualized embeddings.

We choose ELMo as our baseline contextualized embedding model because the word-character lattice model on which MULAN embeddings are built is a nature extension of the sequential LSTM model on which ELMo is based, and therefore it is straightforward to make fair comparison between the two models. Recently, models based on self-attention-network architectures (i.e., Transformer [42]), such as BERT [15] and RoBERTa [67] have been shown to give highly competitive performances compared with ELMo. We take the readily trained Chinese models of BERT[13] and RoBERTa[14] as additional baselines for comparison. However, these models are trained over different sets of data, which makes the comparison indirect.[15] Each contextualized embedding model is used to replace a traditional embedding model for providing input representations to NLP models. We conduct experiments over the following tasks.

POS Tagging. Chinese Treebank [55] 5.0 (CTB5) is selected for joint segmentation and POS tagging tasks. The standard split of the dataset is used following Shao *et al.* [68]. We solve the joint segmentation and tagging problem as character sequence labeling, following the combined label scheme of Ng and Low [69].

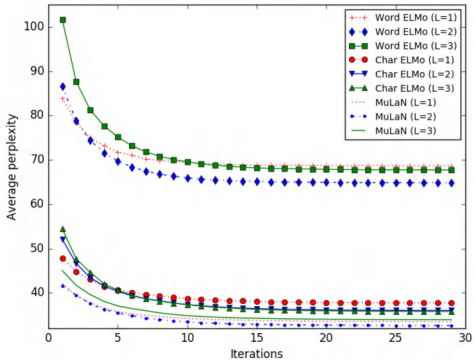


Fig. 8. LM results on development data.

Chunking. Chinese Treebank 4.0 (CTB4) is used. We extract the chunking data following Chen *et al.* [70] using ChunkLinkCTB.[16] The same training, development and test dataset split are used by following Lyu *et al.* [71]. We also solve the joint segmentation and tagging problem as character sequence labeling. In particular, `B`, `I`, `E`, `S` labels are used on characters to represent the beginning, inside, end of a chunk or a single independent character.

NER. Similar to the previous section, we use OntoNotes 4 [49] for named entity recognition, with the `BIOES` tagging scheme [48] being taken on characters.

Parsing. We use Chinese Treebank 5.0 for dependency parsing, the standard unlabeled attachment score (UAS) and labeled attachment score (LAS) are used as evaluation metric.

B. Development Experiments

Comparison between char, word, and lattice based models: Figure 8 shows the averaged forward and backward character perplexity scores on the development data. We find that character-based models give significantly lower perplexities compared to word-based models. This observation is consistent with the findings of Wu *et al.* [72]. One reason can be that word segmentation errors have a major influence on word-level language modeling. In addition, the much larger vocabulary for words makes it more difficult for prediction and generalization. A further disadvantage of word-based language modeling for Chinese arise from different segmentation standards in different treebanks. We thus focus on character and lattice models for the remaining experiments.

Influence of number of layers: In Figure 8, multi-layer models perform better than single-layer models, which shows the stronger representation power of deep hidden states. There is a peak number of layers for each model beyond which further more layers do not lead to better results. Similar to the observation of [13], a two-layer word-based model gives the

[13][Online]. Available: https://github.com/google-research/bert

[14][Online]. Available: https://github.com/pytorch/fairseq/tree/master/examples/roberta

[15]Due to hardware limitation we did not perform training of BERT and RoBERTa models on our dataset.

[16][Online]. Available: https://github.com/rainarch/ChunkLinkCTB

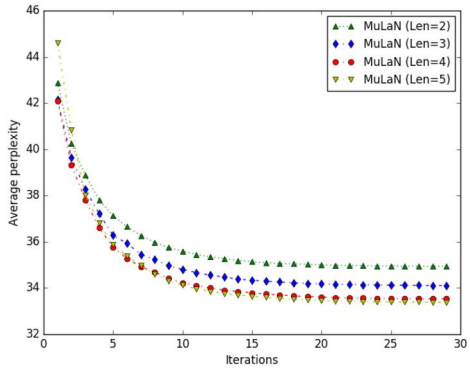


Fig. 9. Influence of max word length for LM.

TABLE XII
LM EVALUATION ON TEST DATA

Model	Forward PPL	Backward PPL
Char ELMo (L=1)	39.10	37.27
Char ELMo (L=2)	36.77	36.15
Char ELMo (L=3)	36.21	36.12
MULAN (L=1)	34.60	33.70
MULAN (L=2)	**33.82**	**32.72**
MULAN (L=3)	35.38	33.83

best per character perplexity result of 64.86. In comparison, a three-layer word-based model gives a worse perplexity of 67.74. For character-based language modeling, the three-layer char-based model gives a 35.81 perplexity, with a 0.58% improvement compared with 36.02 of two-layer char-based model. The best lattice LSTM model gives a perplexity of 33.53, significantly lower compared to that of 35.81 by the best char-based model language model on the development set.

Influence of word length limit: The word length limit len influences the complexity of our model and decides how much word information can be utilized. Figure 9 shows the learning curves of one-layer MULAN with different length limits. Models with $len = 2,3,4,5$ give final perplexities of 34.94, 34.10, 33.53 and 33.40, respectively, showing that the performance benefits from more word information incorporation. However, the average training time per epoch increase from $2h58min$ to $4h41min$, $5h53min$ and $7h35min$ when len increases from 2 to 3, 4 and 5, respectively. Considering that the majority of words do not exceed 4 characters in length, we set $len = 4$ for testing as well as the final models for pre-training.

C. LM Results

Performances on the test dataset are shown in Table XII. For all the models, the forward perplexity is almost the same as the backward perplexity, with the backward values being slightly lower. Consistent with results on the development set, the lattice LSTM model stably outperforms the character-based LSTM models. In particular, the three-layer character-based model gives a 36.21 forward perplexity and a 36.12 backward perplexity, while the two-layer lattice model gives a forward perplexity of 33.82 and a backward perplexity of 32.72, respectively.

TABLE XIII
MULAN RESULTS FOR ON POS TAGGING

Model	POS F
ZPar et al.[74]	93.73
Kurita et al.[75]	94.83
Shao et al.[69]	94.07
Zhang et al.[76]	94.95
Char LSTM-CRF	92.53
Lattice LSTM-CRF	94.86
Lattice LSTM-CRF (ELMo)	95.02
Lattice LSTM-CRF (MULAN)	**95.17**
Lattice LSTM-CRF (BERT)	95.12
Lattice LSTM-CRF (RoBERTa)	95.16

TABLE XIV
MULAN RESULTS FOR CHUNKING. † DENOTES WORD-LEVEL MODELS. ‡ DENOTES WITH GOLD WORD SEGMENTATION

Model	Chunk F
Chen et al.[71]† ‡	91.68
Zhou et al.[77]† ‡	92.11
Lyu et al.[72]†	69.02
Lyu et al.[72]	72.09
Char LSTM-CRF	79.49
Lattice LSTM-CRF	85.39
Lattice LSTM-CRF (ELMo)	85.61
Lattice LSTM-CRF (MULAN)	**85.88**
Lattice LSTM-CRF (BERT)	85.82
Lattice LSTM-CRF (RoBERTa)	85.92

D. Results for Contextualized Embeddings

We compare ELMo embeddings and MULAN embeddings trained on the same dataset for POS-tagging, chunking, NER and dependency parsing.

POS tagging: Results on POS tagging are shown on Table XIII. ZPar [73] employs structured perceptron incorporating local character and word features. Among neural methods, Kurita *et al.* [74] build a joint model for transition-based Chinese syntactic analysis, Shao *et al.* [68] and Zhang *et al.* [75] use BiRNN-CRF and Seq2Seq for joint segmentation and POS tagging, respectively. The above models give the F1-score of 93.73, 94.83, 94.07 and 94.95, respectively. A lattice LSTM-CRF model gives significantly better results compared to char LSTM-CRF. Both ELMo and MULAN embeddings give further improvements to the results, with MULAN giving the best F-score (with word segmentation integrated) of 95.17. Compared with transformer based models, BERT and RoBERTa gives similar results with 95.12 and 95.16. To our knowledge, the result of our model is the current best record in the literature on this dataset.

Chunking: Table XIV shows the results for chunking. With gold segmentation, the word-level chunking models of Chen *et al.* [70] and Zhou *et al.* [76] give F1-scores of 91.68 and 92.11, respectively. In contrast, the performance of the segmentation and chunking pipeline model reported by Lyu [71] gives a significantly lower F1-score of 69.02, demonstrating the negative influence of word segmentation for word-level models. For a character-level model, Lyu *et al.* [71] proposed joint word segmentation, chunking and POS tagging, giving a 72.09 F1-score. Our lattice BiLSTM improves the performance

TABLE XV
MULAN RESULTS FOR NER. † DENOTES WORD-LEVEL MODELS

Model	P	R	F
Wang et al.[59]†	76.43	72.32	74.32
Che et al.[54]†	77.71	72.51	75.02
Yang et al.[60]†	72.98	80.15	76.40
Char LSTM-CRF	74.36	69.43	71.81
Lattice LSTM-CRF	76.35	71.56	73.88
Lattice LSTM-CRF (ELMo)	76.69	75.17	75.92
Lattice LSTM-CRF (MULAN)	**77.05**	**76.77**	**76.91**
Lattice LSTM-CRF (BERT)	77.12	76.02	76.57
Lattice LSTM-CRF (RoBERTa)	76.99	76.74	76.87

TABLE XVI
RESULTS ON DEPENDENCY PARSING

Model	UAS	LAS
Cheng et al.[78]	88.10	85.70
Biaffine LSTM[79]	90.07	85.98
Biaffine LSTM (ELMo)	90.23	88.88
Biaffine LSTM (MULAN)	**90.50**	**89.22**
Biaffine LSTM (BERT)	90.53	89.26
Biaffine LSTM (RoBERTa)	90.56	89.28

with a 85.39 F1-score, demonstrating a significant improvement over a character bi-LSTM model. This also demonstrates the advantage of neural models as compared with statistical models for the task. For all the pretraining models, all embeddings give further improvements, with RoBERTa being the strongest on the dataset, our MULAN model also gives comparable result with 85.88.

NER: Table XV shows the results on NER. Our reimplemented lattice LSTM model achieves a 75.37 F1-score, outperforming the 73.88 F1-score in Table V. This is largely due to the use of embeddings from Song *et al.* [65], which demonstrates the larger word embeddings has a large influence on lattice model performances. Among contextualized embeddings, MULAN representation gives a 76.91 F1-score, outperforming the 75.92 F1-score given by ELMo representations significantly. Our model still better than BERT and RoBERTa in F1-scores, showing the importance of explicit word lexicon information.

Parsing: Table XVI shows the results on dependency parsing. We reimplemented the biaffine parser [78] as our base model and compare the models with different pretraining embeddings. The results are consistent with the other tasks. In particular, the biaffine parser gives results of 90.07 UAS and 85.98 LAS. Using ELMo embeddings, the results are improved to 90.23 UAS and 88.88 LAS. MULAN further led to a small improvement of 90.50 UAS and 89.22 LAS. BERT and RoBERTa gave scores of 90.53/89.26 and 90.56/89.28, respectively, which are slightly better than MULAN. However, as mentioned earlier the results are not directly comparable. MULAN gives highly competitive results among these SAN contextualized embeddings.

Implicit word segmentation: Intuitively, large scale language modeling training can lead to better implicit word segmentation in MULAN. We compare MULAN with the lattice LSTM on a simple segmentation task. In particular, we fix the whole model structure, while adding a trainable softmax layer on each character for predicting `BIES` labels on CTB5. MULAN and lattice LSTM give final F1-scores of 92.00 and 91.18, respectively, demonstrating better word information by MULAN.

TABLE XVII
EXAMPLE OF WORD SEGMENTATION BETWEEN MULAN AND LATTICE LSTM MODELS. BOLD PART INDICATES INCORRECT SEGMENTATION RESULTS

Sentence	新华社北京四月二十日电 Xinhua News Agency, Beijing, April 20th
Gold Seg	新华社 北京 四月 二十日 电 Xinhua Beijing April 20th News
Lattice LSTM	新华社 北京 四月 二十 日电 Xinhua Beijing April **20 NEC Corporation**
MULAN	新华社 北京 四月 二十日 电 Xinhua Beijing April 20th News

A case study is shown in Table XVII. Although both models incorporate word and character information, lattice LSTM makes an incorrect segmentation for the character segment "二十日电" (news on the 20th), largely due to the fact that "日电" (NEC Corporation.) is a named entity in the lexicon. In contrast, MULAN makes the correct prediction for detecting the word "二十日" (20th) and single character "电" (news) thanks to large scale language modeling training.

VIII. CONCLUSION

We discussed the representation of Chinese sentences using a word-character lattice LSTM structure, empirically comparing it with sequential LSTM representations built on character-sequences and pre-segmented word sequences, respectively. On both NER and LM tasks, word-character lattice LSTM gives significantly better performances compared to both the character-based and word-based counterparts. In addition, when being used for contextualized embeddings, the word-character lattice method gives better results compared to its sequential LSTM counterpart, namely ELMO, on a range of Chinese NLP tasks such as POS-tagging, chunking, NER and dependency parsing. We release our code and contextualized embedding models for further research.

REFERENCES

[1] N. Xue, "Chinese word segmentation as character tagging," in *Proc. Int. J. Comput. Linguist. Chin. Lang. Process.*, 2003, pp. 29–48.
[2] Y. Wang, J. Kazama, Y. Tsuruoka, W. Chen, Y. Zhang, and K. Torisawa, "Improving Chinese word segmentation and pos tagging with semi-supervised methods using large auto-analyzed data," in *Proc. 5th Int. Joint Conf. Natural Lang. Process.*, 2011, pp. 309–317.
[3] H. Li, Z. Zhang, Y. Ju, and H. Zhao, "Neural character-level dependency parsing for Chinese," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 5205–5212.
[4] N. Peng and M. Dredze, "Improving named entity recognition for Chinese social media with word segmentation representation learning," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguist.*, 2016, vol. 2, pp. 149–155.
[5] Y. Liu and Y. Zhang, "Unsupervised domain adaptation for joint segmentation and POS-tagging," in *Proc. COLING: Posters*, 2012, pp. 745–754.
[6] W. Jiang, M. Sun, Y. Lü, Y. Yang, and Q. Liu, "Discriminative learning with natural annotations: Word segmentation as a case study," in *Proc. 51st Annu. Meeting Assoc. Comput. Linguist.*, 2013, vol. 1, pp. 761–769.
[7] Y. Liu, Y. Zhang, W. Che, T. Liu, and F. Wu, "Domain adaptation for crf-based Chinese word segmentation using free annotations," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2014, pp. 864–874.

[8] L. Qiu and Y. Zhang, "Word segmentation for Chinese novels," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 2440–2446.

[9] X. Chen, Z. Shi, X. Qiu, and X. Huang, "Adversarial multi-criteria learning for Chinese word segmentation," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguist.*, 2017, vol. 1, pp. 1193–1203.

[10] S. Huang, X. Sun, and H. Wang, "Addressing domain adaptation for Chinese word segmentation with global recurrent structure," in *Proc. 8th Int. Joint Conf. Natural Lang. Process.*, 2017, vol. 1, pp. 184–193.

[11] X. Li, Y. Meng, X. Sun, Q. Han, A. Yuan, and J. Li, "Is word segmentation necessary for deep learning of Chinese representations?" in *Proc. 57th Annu. Meeting Assoc. Comput. Linguist.*, Jul. 2019, pp. 3242–3252.

[12] Z. Dong and Q. Dong, "Hownet-a hybrid language and knowledge resource," in *Proc. IEEE Int. Conf. Natural Lang. Process. Knowl. Eng.*, 2003, pp. 820–824.

[13] M. E. Peters *et al.*, "Deep contextualized word representations," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguist.: Human Lang. Technol.*, 2018, pp. 2227–2237.

[14] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," Tech. Rep., 2018. [Online]. Available: https://openai.com/blog/language-unsupervised/

[15] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguist.: Human Lang. Technol., Volume 1 (Long and Short Papers)*, Jun. 2019, pp. 4171–4186.

[16] Y. Zhang and J. Yang, "Chinese NER using lattice LSTM," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguist. (Volume 1: Long Papers)*, Jul. 2018, pp. 1554–1564.

[17] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," in *Proc. 53rd Annu. Meeting Assoc. Comput. Linguist. 7th Int. Joint Conf. Natural Lang. Process.*, Jul. 2015, pp. 1556–1566.

[18] X. Zhu, P. Sobhani, and H. Guo, "Dag-structured long short-term memory for semantic compositionality," in *Proc. Meeting North Amer. Chapter Assoc. Comput. Linguist.*, 2016, pp. 18–26.

[19] X. Chen, Z. Shi, X. Qiu, and X. Huang, "Dag-based long short-term memory for neural word segmentation," 2017, *arXiv:1707.00248*.

[20] L. Sun, K. Jia, K. Chen, D.-Y. Yeung, B. E. Shi, and S. Savarese, "Lattice long short-term memory for human action recognition," in *Proc. IEEE Int. Conf. Comput. Vision*, Oct. 2017, pp. 2166–2175.

[21] N. Peng, H. Poon, C. Quirk, K. Toutanova, and W. Yih, "Cross-sentence n-ary relation extraction with graph lstms," *Trans. Assoc. Comput. Linguist.*, vol. 5, pp. 101–115, 2017.

[22] M. Sperber, G. Neubig, J. Niehues, and A. Waibel, "Neural lattice-to-sequence models for uncertain inputs," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Sep. 2017, pp. 1380–1389.

[23] J. Su, Z. Tan, D. Xiong, R. Ji, X. Shi, and Y. Liu, "Lattice-based recurrent neural network encoders for neural machine translation," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 3302–3308.

[24] J. Hammerton, "Named entity recognition with long short-term memory," in *Proc. 7th Conf. Natural Lang. Learn. HLT-NAACL*, 2003, pp. 172–175.

[25] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *J. Mach. Learn. Res.*, vol. 12, pp. 2493–2537, 2011.

[26] C. dos Santos, V. Guimaraes, R. Niterói, and R. de Janeiro, "Boosting named entity recognition with neural character embeddings," in *Proc. NEWS 5th Named Entities Workshop*, 2015, pp. 25–33.

[27] Z. Huang, W. Xu, and K. Yu, "Bidirectional LSTM-CRF models for sequence tagging," in *Computer Science*, 2015, *arXiv:1508.01991*.

[28] X. Ma and E. Hovy, "End-to-end sequence labeling via Bi-directional LSTM-CNNs-CRF," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguist*, 2016, vol. 1, pp. 1064–1074.

[29] J. Chiu and E. Nichols, "Named entity recognition with bidirectional LSTM-CNNs," *Trans. Assoc. Comput. Linguist.*, vol. 4, pp. 357–370, 2016.

[30] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguist.: Human Lang. Technol.*, 2016, pp. 260–270.

[31] J. Yang and Y. Zhang, "NCRF++: An open-source neural sequence labeling toolkit," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguist.*, 2018, pp. 74–79. [Online]. Available: http://aclweb.org/anthology/P18-4013

[32] W. Chen, Y. Zhang, and H. Isahara, "Chinese named entity recognition with conditional random fields," in *Proc. 5th SIGHAN Workshop Chin. Lang. Process.*, 2006, pp. 118–121.

[33] Y. Lu, Y. Zhang, and D.-H. Ji, "Multi-prototype Chinese character embedding." in *Proc. 10th Int. Conf. Lang. Resour. Eval.*, 2016, pp. 855–859.

[34] C. Dong, J. Zhang, C. Zong, M. Hattori, and H. Di, "Character-based LSTM-CRF with radical-level features for Chinese named entity recognition," in *Proc. Int. Conf. Comput. Process. Oriental Lang.*, 2016, pp. 239–250.

[35] J. He and H. Wang, "Chinese named entity recognition and word segmentation based on character," in *Proc. 6th SIGHAN Workshop Chin. Lang. Process.*, 2008, pp. 128–132.

[36] Z. Liu, C. Zhu, and T. Zhao, "Chinese named entity recognition with a sequence labeling approach: Based on characters, or based on words?" in *Proc. Adv. Intell. Comput. Theories Appl. With Aspects Artif. Intell.*, 2010, pp. 634–640.

[37] H. Li, M. Hagiwara, Q. Li, and H. Ji, "Comparison of the impact of word segmentation on name tagging for Chinese and Japanese." in *Proc. 9th Int. Conf. Lang. Resour. Eval.*, 2014, pp. 2532–2536.

[38] Z. Jun, G. Jianfeng, C. Eric, and L. Mingjing, "Lexicon optimization for Chinese language modeling," in *Proc. Int. Symp. Chin. Spoken Lang. Process.*, 2000, pp. 283–286.

[39] J. Buckman and G. Neubig, "Neural lattice language models," *Trans. Assoc. Comput. Linguist.*, vol. 6, pp. 529–541, 2018.

[40] A. M. Dai and Q. V. Le, "Semi-supervised sequence learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 3079–3087.

[41] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguist. (Volume 1: Long Papers)*, 2018, pp. 328–339.

[42] A. Vaswani *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., Red Hook, NY, USA: Curran Associates, Inc., 2017, pp. 5998–6008.

[43] X. Chen, X. Qiu, C. Zhu, P. Liu, and X. Huang, "Long short-term memory neural networks for Chinese word segmentation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, Sep. 2015, pp. 1197–1206.

[44] J. Yang, Y. Zhang, and F. Dong, "Neural word segmentation with rich pretraining," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguist.*, 2017, pp. 839–849.

[45] H. Zhao and C. Kit, "Unsupervised segmentation helps supervised learning of character tagging for word segmentation and named entity recognition," in *Proc. 6th SIGHAN Workshop Chin. Lang. Process.*, 2008, pp. 106–111.

[46] L. Liu *et al.*, "Empower sequence labeling with task-aware neural language model," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 5253–5260.

[47] Y. LeCun *et al.*, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, 1989.

[48] L. Ratinov and D. Roth, "Design challenges and misconceptions in named entity recognition," in *Proc. 13th Conf. Comput. Natural Lang. Learn.*, 2009, pp. 147–155.

[49] R. Weischedel *et al.*, "Ontonotes release 4.0," *LDC2011T03, Philadelphia, Penn.: Linguistic Data Consortium*, 2011.

[50] G.-A. Levow, "The third international Chinese language processing bakeoff: Word segmentation and named entity recognition," in *Proc. 5th SIGHAN Workshop Chin. Lang. Process.*, 2006, pp. 108–117.

[51] N. Peng and M. Dredze, "Named entity recognition for Chinese social media with jointly trained embeddings." in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 548–554.

[52] H. He and X. Sun, "F-score driven max margin neural network for named entity recognition in Chinese social media," in *Proc. 15th Conf. Eur. Chapter Assoc. Comput. Linguist.*, 2017, vol. 2, pp. 713–718.

[53] W. Che, M. Wang, C. D. Manning, and T. Liu, "Named entity recognition with bilingual constraints." in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguist.: Human Lang. Technol.*, 2013, pp. 52–62.

[54] J. Yang, Y. Zhang, L. Li, and X. Li, "Yedda: A lightweight collaborative text span annotation tool," in *Proc. ACL Syst. Demonstrations*, 2018, pp. 31–36.

[55] N. Xue, F. Xia, F.-D. Chiou, and M. Palmer, "The Penn Chinese Treebank: Phrase structure annotation of a large corpus," *Natural Lang. Eng.*, vol. 11, no. 02, pp. 207–238, 2005.

[56] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. 26th Int. Conf. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.

[57] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[58] M. Wang, W. Che, and C. D. Manning, "Effective bilingual constraints for semi-supervised learning of named entity recognizers." in *Proc. 27th AAAI Conf. Artif. Intell.*, 2013, pp. 919–925.

[59] J. Yang, Z. Teng, M. Zhang, and Y. Zhang, "Combining discrete and neural features for sequence labeling," in *CICLing*, 2016, pp. 140–154.
[60] A. Chen, F. Peng, R. Shan, and G. Sun, "Chinese named entity recognition with conditional probabilistic models," in *Proc. 5th SIGHAN Workshop Chin. Lang. Process.*, 2006, pp. 173–176.
[61] S. Zhang, Y. Qin, J. Wen, and X. Wang, "Word segmentation and named entity recognition for sighan bakeoff3," in *Proc. 5th SIGHAN Workshop Chinese Lang. Process.*, 2006, pp. 158–161.
[62] J. Zhou, W. Qu, and F. Zhang, "Chinese named entity recognition via joint identification and categorization," *Chin. J. Electron.*, vol. 22, no. 2, pp. 225–230, 2013.
[63] H. He and X. Sun, "A unified model for cross-domain and semi-supervised named entity recognition in Chinese social media." in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 3216–3222.
[64] N. Peng and M. Dredze, "Supplementary results for named entity recognition on Chinese social media with an updated dataset," 2017. [Online]. Available: https://vnpeng.net/papers/golden_horse_supplement.pdf
[65] Y. Song, S. Shi, J. Li, and H. Zhang, "Directional skip-gram: Explicitly distinguishing left and right context for word embeddings," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguist.: Human Lang. Technol., Volume 2 (Short Papers)*, 2018, pp. 175–180.
[66] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent.*, San Diego, CA, USA, May 7–9, 2015.
[67] Y. Liu *et al.*, "RoBERTa: A robustly optimized BERT pretraining approach," 2019, *arXiv:1907.11692*.
[68] Y. Shao, C. Hardmeier, J. Tiedemann, and J. Nivre, "Character-based joint segmentation and POS tagging for Chinese using bidirectional RNN-CRF," in *Proc. 8th Int. Joint Conf. Natural Lang. Process. (Volume 1: Long Papers)*, Nov. 2017, pp. 173–183.
[69] H. T. Ng and J. K. Low, "Chinese part-of-speech tagging: One-at-a-time or all-at-once? word-based or character-based?" in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2004, pp. 277–284. [Online]. Available: http://aclweb.org/anthology/W04-3236
[70] W. Chen, Y. Zhang, and H. Isahara, "An empirical study of Chinese chunking," in *Proc. COLING ACL Main Conf. Poster Sessions*, 2006, pp. 97–104.
[71] C. Lyu, Y. Zhang, and D. Ji, "Joint word segmentation, POS-tagging and syntactic chunking," in *Proc. 30th AAAI Conf. Artif. Intellig.*, 2016, pp. 3007–3014.
[72] W. Wu *et al.*, "Glyce: Glyph-vectors for Chinese character representations," in *Advances in Neural Information Processing Systems*, vol. 32, Curran Associates, Inc. 2019, pp. 2746–2757, [Online]. Available: http://papers.nips.cc/paper/8542-glyce-glyph-vectors-for-chinese-character-representations.pdf.
[73] Y. Zhang and S. Clark, "A fast decoder for joint word segmentation and pos-tagging using a single discriminative model," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2010, pp. 843–852.
[74] S. Kurita, D. Kawahara, and S. Kurohashi, "Neural joint model for transition-based Chinese syntactic analysis," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguist. (Volume 1: Long Papers)*, 2017, pp. 1204–1214.
[75] M. Zhang, N. Yu, and G. Fu, "A simple and effective neural model for joint word segmentation and pos tagging," *IEEE/ACM Trans. Audio, Speech Lang. Process.*, vol. 26, no. 9, pp. 1528–1538, Sep. 2018.
[76] J. Zhou, W. Qu, and F. Zhang, "Exploiting chunk-level features to improve phrase chunking," in *Proc. Joint Conf. Empirical Methods Natural Lang. Process. Comput. Natural Lang. Learn.*, 2012, pp. 557–567.
[77] H. Cheng, H. Fang, X. He, J. Gao, and L. Deng, "Bi-directional attention with agreement for dependency parsing," in *Proc. Conf. Empirical Methods Natural Lang. Process*. Nov. 2016, pp. 2204–2214.
[78] T. Dozat and C. D. Manning, "Deep biaffine attention for neural dependency parsing," in *Proc. 5th Int. Conf. Learn. Representations, Conf. Track*, 2017. [Online]. Available: https://openreview.net/forum?id=Hk95PK9le

Yue Zhang is currently an Associate Professor at Westlake University. He has been working on statistical parsing, text synthesis, natural language synthesis, machine translation, information extraction, sentiment analysis and stock market analysis. His research interests include natural language processing and computational finance. He won the best paper awards of IALP 2017 and COLING 2018. He serves the editorial board for *Transaction of Association of Computational Linguistics* (Action Editor), *ACM Transactions on Asian and Low Resource Language Information Processing* (Associate Editor) and IEEE TRANSACTIONS ON BIG DATA (Associate Editor), and as Area Chairs of COLING 2014/18, NAACL 2015/19, EMNLP 2015/17/19, ACL 2017/18/19. He gave conference tutorials at NAACL 2010, ACL 2014, EMNLP 2016/18.

Yile Wang received the bachelor's and master degree in optical engineering from Zhejiang University in 2013 and 2016, respectively. He is currently working toward the Ph.D. degree in Westlake University currently. From 2016 to 2018, he worked as an Software Development Engineer with Huawei, Hangzhou. His current research interests include language modeling, sequence labeling, and machine learning.

Jie Yang is currently a Postdoctoral Research Fellow at Harvard Medical School and Brigham and Women's Hospital, Harvard University. His research interests include natural language processing, deep learning, artificial intelligence in healthcare and social science. He won the best paper award at COLING 2018, the best demonstration paper nomination at ACL 2018, and ISTD best Ph.D. dissertation award. He serves as a member of the Program Committee at ACL/EMNLP/COLING/NAACL/AAAI/IJCAI, and Scientific Program Committee (SPC) at AMIA 2020. He was awarded as an Outstanding Reviewer at ACL 2018 and COLING 2018.