Chinese NER Using Lattice LSTM

Yue Zhang* and **Jie Yang***
Singapore University of Technology and Design
yue_zhang@sutd.edu.sg
jie_yang@mymail.sutd.edu.sg

Abstract

We investigate a lattice-structured LSTM model for Chinese NER, which encodes a sequence of input characters as well as all potential words that match a lexicon. Compared with character-based methods, our model explicitly leverages word and word sequence information. Compared with word-based methods, lattice LSTM does not suffer from segmentation errors. Gated recurrent cells allow our model to choose the most relevant characters and words from a sentence for better NER results. Experiments on various datasets show that lattice LSTM outperforms both word-based and character-based LSTM baselines, achieving the best results.

1 Introduction

As a fundamental task in information extraction, named entity recognition (NER) has received constant research attention over the recent years. The task has traditionally been solved as a sequence labeling problem, where entity boundary and category labels are jointly predicted. The current state-of-the-art for English NER has been achieved by using LSTM-CRF models (Lample et al., 2016; Ma and Hovy, 2016; Chiu and Nichols, 2016; Liu et al., 2018) with character information being integrated into word representations.

Chinese NER is correlated with word segmentation. In particular, named entity boundaries are also word boundaries. One intuitive way of performing Chinese NER is to perform word segmentation first, before applying word sequence labeling. The segmentation → NER pipeline, however, can suffer the potential issue of error propagation, since NEs are an important source of OOV in segmentation, and incorrectly segmented entity boundaries lead to NER errors. This problem can be severe in the open domain since cross-domain word segmentation remains an unsolved problem (Liu and Zhang, 2012; Jiang et al., 2013; Liu et al., 2014; Qiu and Zhang, 2015; Chen et al., 2017; Huang et al., 2017). It has been shown that character-based methods outperform word-based methods for Chinese NER (He and Wang, 2008; Liu et al., 2010; Li et al., 2014).

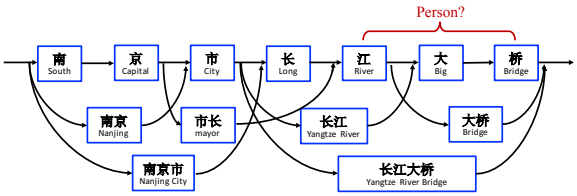


Figure 1: Word character lattice.

One drawback of character-based NER, however, is that explicit word and word sequence information is not fully exploited, which can be potentially useful. To address this issue, we integrate latent word information into character-based LSTM-CRF by representing lexicon words from the sentence using a lattice structure LSTM. As shown in Figure 1, we construct a word-character lattice by matching a sentence with a large automatically-obtained lexicon. As a result, 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)".

Since there are an exponential number of word-character paths in a lattice, 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

* Equal contribution.

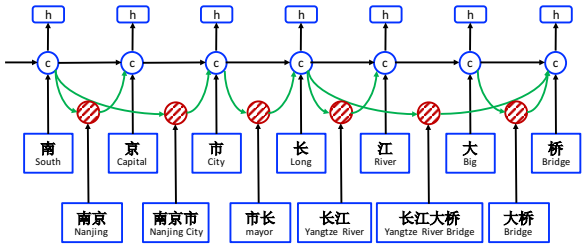


Figure 2: Lattice LSTM structure.

different paths to each character. Trained over NER data, the lattice LSTM can learn to find more useful words from context automatically for better NER performance. Compared with character-based and word-based NER methods, our model has the advantage of leveraging explicit word information over character sequence labeling without suffering from segmentation error.

Results 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. Our code and data are released at `https://github.com/jiesutd/LatticeLSTM`.

2 Related Work

Our work is in line with existing methods using neural network for NER. Hammerton (2003) attempted to solve the problem using a unidirectional LSTM, which was among the first neural models for NER. Collobert et al. (2011) used a CNN-CRF structure, obtaining competitive results to the best statistical models. dos Santos et al. (2015) used character CNN to augment a CNN-CRF model. Most recent work leverages an LSTM-CRF architecture. Huang et al. (2015) uses hand-crafted spelling features; Ma and Hovy (2016) and Chiu and Nichols (2016) use a character CNN to represent spelling characteristics; Lample et al. (2016) use a character LSTM instead. Our baseline word-based system takes a similar structure to this line of work.

Character sequence labeling has been the dominant approach for Chinese NER (Chen et al., 2006b; Lu et al., 2016; Dong et al., 2016). 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 (He and Wang, 2008; Liu et al., 2010; Li et al., 2014). 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.

How to better leverage word information for Chinese NER has received continued research attention (Gao et al., 2005), where segmentation information has been used as soft features for NER (Zhao and Kit, 2008; Peng and Dredze, 2015; He and Sun, 2017a), and joint segmentation and NER has been investigated using dual decomposition (Xu et al., 2014), multi-task learning (Peng and Dredze, 2016), etc. Our work is in line, focusing on neural representation learning. While the above methods can be affected by segmented training data and segmentation errors, our method does not require a word segmentor. The model is conceptually simpler by not considering multi-task settings.

External sources of information has been leveraged for NER. In particular, lexicon features have been widely used (Collobert et al., 2011; Passos et al., 2014; Huang et al., 2015; Luo et al., 2015). Rei (2017) uses a word-level language modeling objective to augment NER training, performing multi-task learning over large raw text. Peters et al. (2017) pretrain a character language model to enhance word representations. Yang et al. (2017b) exploit cross-domain and cross-lingual knowledge via multi-task learning. We leverage external data by pretraining word embedding lexicon over large automatically-segmented texts, while semi-supervised techniques such as language modeling are orthogonal to and can also be used for our lattice LSTM model.

Lattice structured RNNs can be viewed as a natural extension of tree-structured RNNs (Tai et al., 2015) to DAGs. They have been used to model motion dynamics (Sun et al., 2017), dependency-discourse DAGs (Peng et al., 2017), as well as speech tokenization lattice (Sperber et al., 2017) and multi-granularity segmentation outputs (Su et al., 2017) 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, and the first to use a word-character lattice for segmentation-free Chinese NER.

3 Model

We follow the best English NER model (Huang et al., 2015; Ma and Hovy, 2016; Lample et al., 2016), using LSTM-CRF as the main network structure. 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 (Ratinov and Roth, 2009) for both word-based and character-based NER tagging.

3.1 Character-Based Model

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

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

$\mathbf{e}^c$ 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:

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

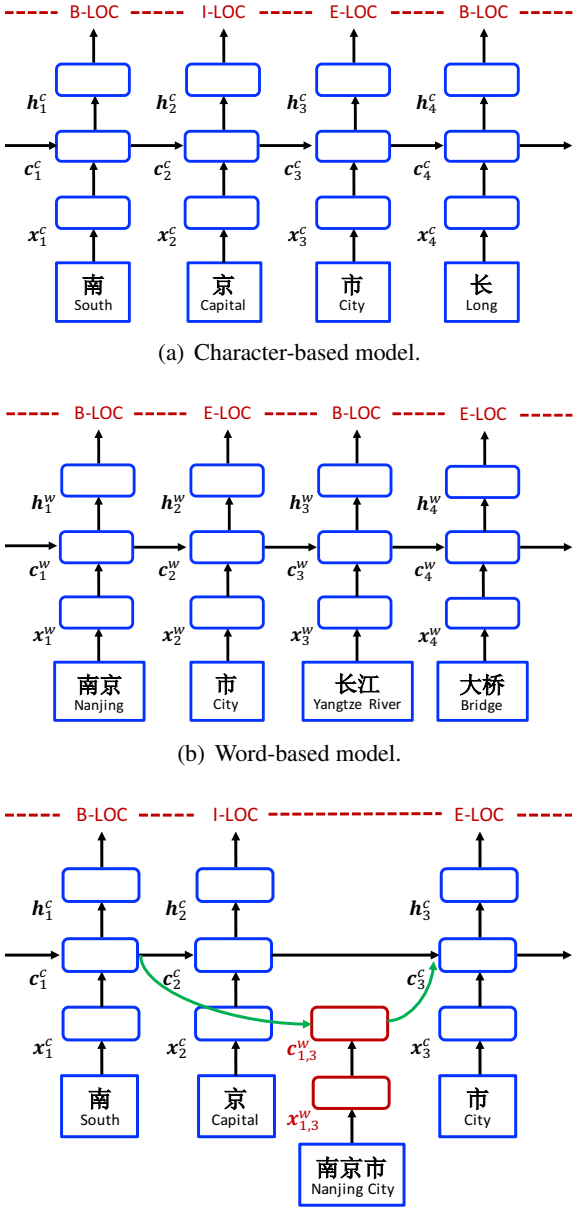
A standard CRF model (Eq. 17) is used on $\mathbf{h}_1^c, \mathbf{h}_2^c, \ldots, \mathbf{h}_m^c$ for sequence labelling.

• **Char + bichar**. Character bigrams have been shown useful for representing characters in word segmentation (Chen et al., 2015; Yang et al., 2017a). We augment the character-based model with bigram information by concatenating bigram embeddings with character embeddings:

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

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 NER models can lead to improved performance (Zhao and Kit, 2008; Peng and Dredze, 2016).



(a) Character-based model.

(b) Word-based model.

(c) Lattice model.

Figure 3: Models.[1]

We augment the character representation with segmentation information by concatenating segmentation label embeddings to character embeddings:

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

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 repre-

[1] 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.

senting segmentation (Xue, 2003).

$$\mathbf{h}_i^w = [\overrightarrow{\mathbf{h}_i^w}; \overleftarrow{\mathbf{h}_i^w}] \quad (5)$$

Similar to the character-based case, a standard CRF model (Eq. 17) is used on $\mathbf{h}_1^w, \mathbf{h}_2^w, \ldots, \mathbf{h}_m^w$ for sequence labelling.

3.2 Word-Based Model

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:

$$\mathbf{x}_i^w = \mathbf{e}^w(w_i), \quad (6)$$

where $\mathbf{e}^w$ denotes a word embedding lookup table. A bi-directioanl 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:

Integrating character representations

Both character CNN (Ma and Hovy, 2016) and LSTM (Lample et al., 2016) have been used for representing the character sequence within a word. We experiment with both for Chinese NER. 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 $\mathbf{x}_i^c$:

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

• **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 $\overrightarrow{\mathbf{h}}^c_{t(i,1)}, \ldots, \overrightarrow{\mathbf{h}}^c_{t(i,len(i))}$ and $\overleftarrow{\mathbf{h}}^c_{t(i,1)}, \ldots, \overleftarrow{\mathbf{h}}^c_{t(i,len(i))}$ 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:

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

• **Word + char LSTM′**. We investigate a variation of word + char LSTM model that uses a **single** LSTM to obtain $\overrightarrow{\mathbf{h}}^c_j$ and $\overleftarrow{\mathbf{h}}^c_j$ for each c_j. It is similar with the structure of Liu et al. (2018) 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 (LeCun et al., 1989) 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:

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

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

3.3 Lattice Model

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 $\mathbb{D}$. As indicated in Section 2, we use automatically segmented large raw text for buinding $\mathbb{D}$. 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 chacracter 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 CRF sequence labelling using Eq. 17.

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

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

where $\mathbf{e}^w$ denotes the same word embedding lookup table as in Section 3.2.

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:

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

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).[2] 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 $\mathbf{c}_{b,e}^c$:

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

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

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

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 $\mathbf{1}$.

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

The final hidden vectors $\mathbf{h}_j^c$ are still computed as described by Eq. 11. During NER 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 NER labelling.

[2] 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.

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

Table 1: Statistics of datasets.

3.4 Decoding and Training

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

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

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

4 Experiments

We carry out an extensive set of experiments to investigate the effectiveness of word-character lattice LSTMs 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.

4.1 Experimental Settings

Data. Four datasets are used in this paper, which include OntoNotes 4 (Weischedel et al., 2011), MSRA (Levow, 2006) Weibo NER (Peng and

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 2: Detailed statistics of resume NER.

Dredze, 2015; He and Sun, 2017a) and a Chinese resume dataset that we annotate. Statistics of the datasets are shown in Table 1. We take the same data split as Che et al. (2013) 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.[3]

For more variety in test domains, we collected a resume dataset from Sina Finance[4], 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. Statistics of the dataset is shown in Table 2. 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 word segmentor of Yang et al. (2017a) 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 Yang et al. (2017a) off the shelf[5], which is trained using CTB 6.0 (Xue et al., 2005).

[3] https://www.weibo.com/

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

[5] https://github.com/jiesutd/RichWordSegmentor

Parameter	Value	Parameter	Value
char emb size	50	bigram emb size	50
lattice emb size	50	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

Table 3: Hyper-parameter values.

Word Embeddings. We pretrain word embeddings using word2vec (Mikolov et al., 2013) over automatically segmented Chinese Giga-Word[6], obtaining 704.4k words in a final lexicon. In particular, the number of single-character, two-character and three-character words are 5.7k, 291.5k, 278.1k, 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 3 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 (Srivastava et al., 2014) 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.

4.2 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 4, 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 3.1 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

[6] https://catalog.ldc.upenn.edu/LDC2011T13

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**

Table 4: Development results.

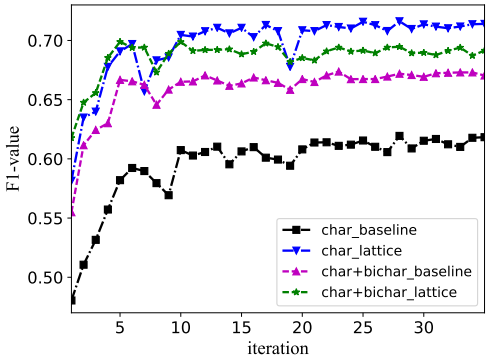


Figure 4: F1 against training iteration number.

Input	Models	P	R	F1
Gold seg	Yang et al. (2016)	65.59	71.84	68.57
	Yang et al. (2016)*†	72.98	**80.15**	**76.40**
	Che et al. (2013)*	77.71	72.51	75.02
	Wang et al. (2013)*	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**

Table 5: Main results on OntoNotes.

among various character representations. We thus choose this model in the remaining experiments.

Word-based NER. Table 4 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 3.2, 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.

Lattice-based NER. Figure 4 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 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 4, the lattice LSTM-CRF model gives a development F1-score of 71.62%, which is significantly[7] 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.

4.3 Final Results

OntoNotes. The OntoNotes test results are shown in Table 5[8]. With gold-standard segmentation, our word-based methods give competitive results to the state-of-the-art on the dataset (Che et al., 2013; Wang et al., 2013), 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

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

[8]In Table 5, 6 and 7, we use * to denote a model with external labeled data for semi-supervised learning. † means that the model also uses discrete features.

Models	P	R	F1
Chen et al. (2006a)	91.22	81.71	86.20
Zhang et al. (2006)*	92.20	90.18	91.18
Zhou et al. (2013)	91.86	88.75	90.28
Lu et al. (2016)	–	–	87.94
Dong et al. (2016)	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 6: Main results on MSRA.

Models	NE	NM	Overall
Peng and Dredze (2015)	51.96	61.05	56.05
Peng and Dredze (2016)*	**55.28**	**62.97**	**58.99**
He and Sun (2017a)	50.60	59.32	54.82
He and Sun (2017b)*	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 7: Weibo NER results.

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 6. 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 (Chen et al., 2006a; Zhang et al., 2006; Zhou et al., 2013) and character embedding features (Lu et al., 2016). Dong et al. (2016) exploit neural LSTM-CRF with radical features.

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 7, 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 Peng and Dredze (2016) and He and Sun (2017b), who explore rich embedding features, cross-domain and semi-supervised data, some of which are orthogonal to our model[9].

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**

Table 8: Main results on resume NER.

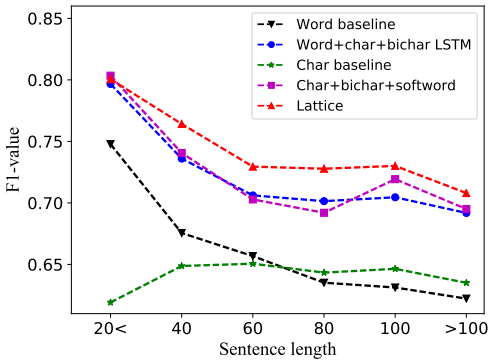


Figure 5: F1 against sentence length.

Results on the resume NER test data are shown in Table 8. 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.

4.4 Discussion

F1 against sentence length. Figure 5 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 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

[9]The results of Peng and Dredze (2015, 2016) are taken from Peng and Dredze (2017).

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$_{ORG}$ ...

Table 9: Example. Red and green represent incorrect and correct entities, respectively.

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

Table 10: Entities in lexicon.

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 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 9 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" 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.

Entities in lexicon. Table 10 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 (Ratinov and Roth, 2009; Chiu and Nichols, 2016), 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.

5 Conclusion

We empirically investigated a lattice LSTM-CRF representations for Chinese NER, finding that it gives consistently superior performance compared to word-based and character-based LSTM-CRF across different domains. The lattice method is fully independent of word segmentation, yet more effective in using word information thanks to the freedom of choosing lexicon words in a context for NER disambiguation.

Acknowledgments

We thank the anonymous reviewers for their insightful comments.

References

Wanxiang Che, Mengqiu Wang, Christopher D Manning, and Ting Liu. 2013. Named entity recognition with bilingual constraints. In *HLT-NAACL*. pages 52–62.

Aitao Chen, Fuchun Peng, Roy Shan, and Gordon Sun. 2006a. Chinese named entity recognition with conditional probabilistic models. In *Proceedings of the*

Fifth SIGHAN Workshop on Chinese Language Processing. pages 173–176.

Wenliang Chen, Yujie Zhang, and Hitoshi Isahara. 2006b. Chinese named entity recognition with conditional random fields. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*. pages 118–121.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. 2015. Long short-term memory neural networks for chinese word segmentation. In *EMNLP*. Lisbon, Portugal, pages 1197–1206. http://aclweb.org/anthology/D15-1141.

Xinchi Chen, Zhan Shi, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-criteria learning for Chinese word segmentation. In *ACL*. volume 1, pages 1193–1203.

Jason Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *TACL* 4:357–370. https://transacl.org/ojs/index.php/tacl/article/view/792.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12(Aug):2493–2537.

Chuanhai Dong, Jiajun Zhang, Chengqing Zong, Masanori Hattori, and Hui Di. 2016. Character-based LSTM-CRF with radical-level features for Chinese named entity recognition. In *International Conference on Computer Processing of Oriental Languages*. Springer, pages 239–250.

Cıcero dos Santos, Victor Guimaraes, RJ Niterói, and Rio de Janeiro. 2015. Boosting named entity recognition with neural character embeddings. In *Proceedings of NEWS 2015 The Fifth Named Entities Workshop*. page 25.

Jianfeng Gao, Mu Li, Chang-Ning Huang, and Andi Wu. 2005. Chinese word segmentation and named entity recognition: A pragmatic approach. *Computational Linguistics* 31(4):531–574.

James Hammerton. 2003. Named entity recognition with long short-term memory. In *HLT-NAACL 2003-Volume 4*. pages 172–175.

Hangfeng He and Xu Sun. 2017a. F-score driven max margin neural network for named entity recognition in Chinese social media. In *EACL*. volume 2, pages 713–718.

Hangfeng He and Xu Sun. 2017b. A unified model for cross-domain and semi-supervised named entity recognition in Chinese social media. In *AAAI*. pages 3216–3222.

Jingzhou He and Houfeng Wang. 2008. Chinese named entity recognition and word segmentation based on character. In *Proceedings of the Sixth SIGHAN Workshop on Chinese Language Processing*.

Shen Huang, Xu Sun, and Houfeng Wang. 2017. Addressing domain adaptation for chinese word segmentation with global recurrent structure. In *IJCNLP*. volume 1, pages 184–193.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991* .

Wenbin Jiang, Meng Sun, Yajuan Lü, Yating Yang, and Qun Liu. 2013. Discriminative learning with natural annotations: Word segmentation as a case study. In *ACL*. volume 1, pages 761–769.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *NAACL-HLT*. pages 260–270.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation* 1(4):541–551.

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

Haibo Li, Masato Hagiwara, Qi Li, and Heng Ji. 2014. Comparison of the impact of word segmentation on name tagging for Chinese and japanese. In *LREC*. pages 2532–2536.

Liyuan Liu, Jingbo Shang, Frank Xu, Xiang Ren, Huan Gui, Jian Peng, and Jiawei Han. 2018. Empower sequence labeling with task-aware neural language model. *AAAI* .

Yang Liu and Yue Zhang. 2012. Unsupervised domain adaptation for joint segmentation and pos-tagging. *Proceedings of COLING 2012: Posters* pages 745–754.

Yijia Liu, Yue Zhang, Wanxiang Che, Ting Liu, and Fan Wu. 2014. Domain adaptation for crf-based Chinese word segmentation using free annotations. In *EMNLP*. pages 864–874.

Zhangxun Liu, Conghui Zhu, and Tiejun Zhao. 2010. Chinese named entity recognition with a sequence labeling approach: based on characters, or based on words? In *Advanced intelligent computing theories and applications. With aspects of artificial intelligence*, Springer, pages 634–640.

Yanan Lu, Yue Zhang, and Dong-Hong Ji. 2016. Multi-prototype Chinese character embedding. In *LREC*.

Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. 2015. Joint entity recognition and disambiguation. In *EMNLP*. pages 879–888.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via Bi-directional LSTM-CNNs-CRF. In *ACL*. volume 1, pages 1064–1074.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. pages 3111–3119.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. In *CoNLL*. pages 78–86.

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*. volume 2, pages 149–155.

Nanyun Peng and Mark Dredze. 2017. Supplementary results for named entity recognition on chinese social media with an updated dataset .

Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. 2017. Cross-sentence n-ary relation extraction with graph lstms. *TACL* 5:101–115.

Matthew Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *ACL*. volume 1, pages 1756–1765.

Likun Qiu and Yue Zhang. 2015. Word segmentation for chinese novels. In *AAAI*. pages 2440–2446.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *CoNLL*. pages 147–155.

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *ACL*. volume 1, pages 2121–2130.

Matthias Sperber, Graham Neubig, Jan Niehues, and Alex Waibel. 2017. Neural lattice-to-sequence models for uncertain inputs. In *EMNLP*. pages 1380–1389. https://www.aclweb.org/anthology/D17-1145.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *JMLR* 15(1):1929–1958.

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.

Lin Sun, Kui Jia, Kevin Chen, Dit-Yan Yeung, Bertram E. Shi, and Silvio Savarese. 2017. Lattice long short-term memory for human action recognition. In *ICCV*.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *ACL-IJCNLP*. Beijing, China, pages 1556–1566. http://www.aclweb.org/anthology/P15-1150.

Mengqiu Wang, Wanxiang Che, and Christopher D Manning. 2013. Effective bilingual constraints for semi-supervised learning of named entity recognizers. In *AAAI*.

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

Y Xu, Y Wang, T Liu, J Liu, Y Fan, Y Qian, J Tsujii, and EI Chang. 2014. Joint segmentation and named entity recognition using dual decomposition in Chinese discharge summaries. *JAMIA* 21(e1):e84–92.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Marta Palmer. 2005. The penn Chinese treebank: Phrase structure annotation of a large corpus. *Natural language engineering* 11(02):207–238.

Nianwen Xue. 2003. Chinese word segmentation as character tagging. In *International Journal of Computational Linguistics and Chinese Language Processing*.

Jie Yang, Zhiyang Teng, Meishan Zhang, and Yue Zhang. 2016. Combining discrete and neural features for sequence labeling. In *International Conference on Computational Linguistics and Intelligent Text Processing*.

Jie Yang, Yue Zhang, and Fei Dong. 2017a. Neural word segmentation with rich pretraining. In *ACL*. Vancouver, Canada, pages 839–849. http://aclweb.org/anthology/P17-1078.

Zhilin Yang, Ruslan Salakhutdinov, and William W Cohen. 2017b. Transfer learning for sequence tagging with hierarchical recurrent networks. In *ICLR*.

Suxiang Zhang, Ying Qin, Juan Wen, and Xiaojie Wang. 2006. Word segmentation and named entity recognition for sighan bakeoff3. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*. pages 158–161.

Hai Zhao and Chunyu Kit. 2008. Unsupervised segmentation helps supervised learning of character tagging for word segmentation and named entity recognition. In *Proceedings of the Sixth SIGHAN Workshop on Chinese Language Processing*.

Junsheng Zhou, Weiguang Qu, and Fen Zhang. 2013. Chinese named entity recognition via joint identification and categorization. *Chinese Journal of Electronics* 22(2):225–230.