Long Short-Term Memory Neural Networks for Chinese Word Segmentation

Xinchi Chen, Xipeng Qiu[*], Chenxi Zhu, Pengfei Liu, Xuanjing Huang
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai, China
{xinchichen13,xpqiu,czhu13,pfliu14,xjhuang}@fudan.edu.cn

Abstract

Currently most of state-of-the-art methods for Chinese word segmentation are based on supervised learning, whose features are mostly extracted from a local context. These methods cannot utilize the long distance information which is also crucial for word segmentation. In this paper, we propose a novel neural network model for Chinese word segmentation, which adopts the long short-term memory (LSTM) neural network to keep the previous important information in memory cell and avoids the limit of window size of local context. Experiments on PKU, MSRA and CTB6 benchmark datasets show that our model outperforms the previous neural network models and state-of-the-art methods.

1 Introduction

Word segmentation is a fundamental task for Chinese language processing. In recent years, Chinese word segmentation (CWS) has undergone great development. The popular method is to regard word segmentation task as a sequence labeling problem (Xue, 2003; Peng et al., 2004). The goal of sequence labeling is to assign labels to all elements in a sequence, which can be handled with supervised learning algorithms such as Maximum Entropy (ME) (Berger et al., 1996) and Conditional Random Fields (CRF) (Lafferty et al., 2001). However, the ability of these models is restricted by the design of features, and the number of features could be so large that the result models are too large for practical use and prone to overfit on training corpus.

Recently, neural network models have increasingly used for NLP tasks for their ability to minimize the effort in feature engineering (Collobert et al., 2011; Socher et al., 2013; Turian et al., 2010; Mikolov et al., 2013b; Bengio et al., 2003). Collobert et al. (2011) developed the SENNA system that approaches or surpasses the state-of-the-art systems on a variety of sequence labeling tasks for English. Zheng et al. (2013) applied the architecture of Collobert et al. (2011) to Chinese word segmentation and POS tagging, also he proposed a perceptron style algorithm to speed up the training process with negligible loss in performance. Pei et al. (2014) models tag-tag interactions, tag-character interactions and character-character interactions based on Zheng et al. (2013). Chen et al. (2015) proposed a gated recursive neural network (GRNN) to explicitly model the combinations of the characters for Chinese word segmentation task. Each neuron in GRNN can be regarded as a different combination of the input characters. Thus, the whole GRNN has an ability to simulate the design of the sophisticated features in traditional methods.

Despite of their success, a limitation of them is that their performances are easily affected by the size of the context window. Intuitively, many words are difficult to segment based on the local information only. For example, the segmentation of the following sentence needs the information of the long distance collocation.

冬天 (winter)，能 (can) 穿 (wear) 多少 (amount) 穿 (wear) 多少 (amount)；夏天 (summer)，能 (can) 穿 (wear) 多 (more) 少 (little) 穿 (wear) 多 (more) 少 (little)。

Without the word “夏天 (summer)” or “冬天 (winter)”, it is difficult to segment the phrase “能穿多少穿多少”. Therefore, we usually need utilize the non-local information for more accurate word segmentation. However, it does not work by simply increasing the context window size. As reported in (Zheng et al., 2013), the performance drops smoothly when the window size is larger than 3. The reason is that the number of its parameters is so large that the trained network has

[*]Corresponding author.

overfitted on training data. Therefore, it is necessary to capture the potential long-distance dependencies without increasing the size of the context window.

In order to address this problem, we propose a neural model based on Long Short-Term Memory Neural Network (LSTM) (Hochreiter and Schmidhuber, 1997) that explicitly model the previous information by exploiting input, output and forget gates to decide how to utilize and update the memory of pervious information. Intuitively, if the LSTM unit detects an important feature from an input sequence at early stage, it easily carries this information (the existence of the feature) over a long distance, hence, capturing the potential useful long-distance information. We evaluate our model on three popular benchmark datasets (PKU, MSRA and CTB6), and the experimental results show that our model achieves the state-of-the-art performance with the smaller context window size (0,2).

The contributions of this paper can be summarized as follows.

- We first introduce the LSTM neural network for Chinese word segmentation. The LSTM can capture potential long-distance dependencies and keep the previous useful information in memory, which avoids the limit of the size of context window.
- Although there are relatively few researches of applying dropout method to the LSTM, we investigate several dropout strategies and find that dropout is also effective to avoid the overfitting of the LSTM.
- Despite Chinese word segmentation being a specific case, our model can be easily generalized and applied to the other sequence labeling tasks.

2 Neural Model for Chinese Word Segmentation

Chinese word segmentation is usually regarded as character-based sequence labeling. Each character is labeled as one of {B, M, E, S} to indicate the segmentation. {B, M, E} represent *Begin*, *Middle*, *End* of a multi-character segmentation respectively, and S represents a *Single* character segmentation.

The neural model is usually characterized by three specialized layers: (1) a character embedding layer; (2) a series of classical neural network layers and (3) tag inference layer. An illustration is shown in Figure 1.

The most common tagging approach is based on a local window. The window approach assumes that the tag of a character largely depends on its neighboring characters. Given an input sentence $c^{(1:n)}$, a window of size k slides over the sentence from character $c^{(1)}$ to $c^{(n)}$, where n is the length of the sentence. As shown in Figure 1, for each character $c^{(t)}(1 \leq t \leq n)$, the context characters ($c^{(t-2)}$,$c^{(t-1)}$,$c^{(t)}$,$c^{(t+1)}$,$c^{(t+2)}$) are fed into the lookup table layer when the window size k is 5. The characters exceeding the sentence boundaries are mapped to one of two special symbols, namely "start" and "end" symbols. The character embeddings extracted by the lookup table layer are then concatenated into a single vector $\mathbf{x}^{(t)} \in \mathbb{R}^{H_1}$, where $H_1 = k \times d$ is the size of layer 1. Then $\mathbf{x}^{(t)}$ is fed into the next layer which performs linear transformation followed by an element-wise activation function g such as sigmoid function $\sigma(x) = (1+e^{-x})^{-1}$ and hyperbolic tangent function $\phi(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$here.

$$\mathbf{h}^{(t)} = g(\mathbf{W}_1\mathbf{x}^{(t)} + \mathbf{b}_1), \tag{1}$$

where $\mathbf{W}_1 \in \mathbb{R}^{H_2 \times H_1}$, $\mathbf{b}_1 \in \mathbb{R}^{H_2}$, $\mathbf{h}^{(t)} \in \mathbb{R}^{H_2}$. H_2 is a hyper-parameter which indicates the number of hidden units in layer 2. Given a set of tags $\mathcal{T}$ of size $|\mathcal{T}|$, a similar linear transformation is performed except that no non-linear function is followed:

$$\mathbf{y}^{(t)} = \mathbf{W}_2\mathbf{h}^{(t)} + \mathbf{b}_2, \tag{2}$$

where $\mathbf{W}_2 \in \mathbb{R}^{|\mathcal{T}| \times H_2}$, $\mathbf{b}_2 \in \mathbb{R}^{|\mathcal{T}|}$. $\mathbf{y}^{(t)} \in \mathbb{R}^{|\mathcal{T}|}$ is the score vector for each possible tag. In Chinese word segmentation, the most prevalent tag set $T \subseteq \mathcal{T}$ is {B, M, E, S} as mentioned above.

To model the tag dependency, a transition score $\mathbf{A}_{ij}$ is introduced to measure the probability of jumping from tag $i \in T$ to tag $j \in T$ (Collobert et al., 2011). Although this model works well for Chinese word segmentation and other sequence labeling tasks, it just utilizes the information of context of a limited-length window. Some useful long distance information is neglected.

3 Long Short-Term Memory Neural Network for Chinese Word Segmentation

In this section, we introduce the LSTM neural network for Chinese word segmentation.

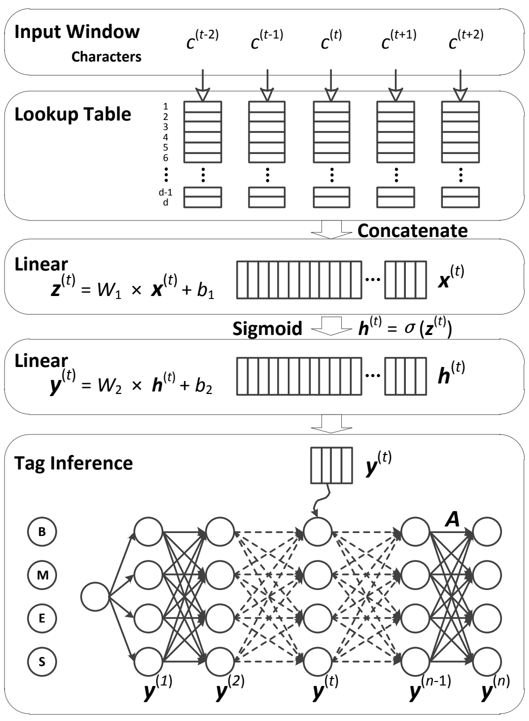

Input Window
Characters
$c^{(t-2)}$ $c^{(t-1)}$ $c^{(t)}$ $c^{(t+1)}$ $c^{(t+2)}$
Lookup Table
Concatenate
Linear
$z^{(t)} = W_1 \times x^{(t)} + b_1$
$x^{(t)}$
Sigmoid
$h^{(t)} = \sigma(z^{(t)})$
Linear
$y^{(t)} = W_2 \times h^{(t)} + b_2$
$h^{(t)}$
Tag Inference
$y^{(t)}$
A
B M E S
$y^{(1)}$ $y^{(2)}$ $y^{(t)}$ $y^{(n-1)}$ $y^{(n)}$

Figure 1: General architecture of neural model for Chinese word segmentation.

3.1 Character Embeddings

The first step of using neural network to process symbolic data is to represent them into distributed vectors, also called embeddings (Bengio et al., 2003; Collobert and Weston, 2008).

Formally, in Chinese word segmentation task, we have a character dictionary $\mathcal{C}$ of size $|\mathcal{C}|$. Unless otherwise specified, the character dictionary is extracted from the training set and unknown characters are mapped to a special symbol that is not used elsewhere. Each character $c \in \mathcal{C}$ is represented as a real-valued vector (character embedding) $\mathbf{v}_c \in \mathbb{R}^d$ where d is the dimensionality of the vector space. The character embeddings are then stacked into an embedding matrix $\mathbf{M} \in \mathbb{R}^{d \times |\mathcal{C}|}$. For a character $c \in \mathcal{C}$, the corresponding character embedding $\mathbf{v}_c \in \mathbb{R}^d$ is retrieved by the lookup table layer. And the lookup table layer can be regarded as a simple projection layer where the character embedding for each context character is achieved by table lookup operation according to its index.

3.2 LSTM

The long short term memory neural network (LSTM) (Hochreiter and Schmidhuber, 1997) is an extension of the recurrent neural network (RNN).

The RNN has recurrent hidden states whose output at each time is dependent on that of the previous time. More formally, given a sequence $\mathbf{x}^{(1:n)} = (\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(t)}, \ldots, \mathbf{x}^{(n)})$, the RNN updates its recurrent hidden state $\mathbf{h}^{(t)}$ by

$$\mathbf{h}^{(t)} = g(\mathbf{U}\mathbf{h}^{(t-1)} + \mathbf{W}\mathbf{x}^{(t)} + \mathbf{b}), \tag{3}$$

where g is a nonlinear function as mentioned above.

Though RNN has been proven successful on many tasks such as speech recognition (Vinyals et al., 2012), language modeling (Mikolov et al., 2010) and text generation (Sutskever et al., 2011), it can be difficult to train them to learn long-term dynamics, likely due in part to the vanishing and exploding gradient problem (Hochreiter and Schmidhuber, 1997).

The LSTM provides a solution by incorporating memory units that allow the network to learn when to forget previous information and when to update the memory cells given new information. Thus, it is a natural choice to apply LSTM neural network to word segmentation task since the LSTM neural network can learn from data with long range temporal dependencies (memory) due to the considerable time lag between the inputs and their corresponding outputs. In addition, the LSTM has been applied successfully in many NLP tasks, such as text classification (Liu et al., 2015) and machine translation (Sutskever et al., 2014).

The core of the LSTM model is a memory cell $\mathbf{c}$ encoding memory at every time step of what inputs have been observed up to this step (see Figure 2) . The behavior of the cell is controlled by three "gates", namely input gate $\mathbf{i}$, forget gate $\mathbf{f}$ and output gate $\mathbf{o}$. The operations on gates are defined as element-wise multiplications, thus gate can either scale the input value if the gate is non-zero vector or omit input if the gate is zero vector. The output of output gate will be fed into the next time step $t+1$ as previous hidden state and input of upper layer of neural network at current time step t. The definitions of the gates, cell update and output are as follows:

$$\mathbf{i}^{(t)} = \sigma(\mathbf{W}_{ix}\mathbf{x}^{(t)} + \mathbf{W}_{ih}\mathbf{h}^{(t-1)} + \mathbf{W}_{ic}\mathbf{c}^{(t-1)}), \tag{4}$$

$$\mathbf{f}^{(t)} = \sigma(\mathbf{W}_{fx}\mathbf{x}^{(t)} + \mathbf{W}_{fh}\mathbf{h}^{(t-1)} + \mathbf{W}_{fc}\mathbf{c}^{(t-1)}), \tag{5}$$

$$\mathbf{c}^{(t)} = \mathbf{f}^{(t)} \odot \mathbf{c}^{(t-1)} + \mathbf{i}^{(t)} \odot \phi(\mathbf{W}_{cx}\mathbf{x}^{(t)} + \mathbf{W}_{ch}\mathbf{h}^{(t-1)}), \tag{6}$$

$$\mathbf{o}^{(t)} = \sigma(\mathbf{W}_{ox}\mathbf{x}^{(t)} + \mathbf{W}_{oh}\mathbf{h}^{(t-1)} + \mathbf{W}_{oc}\mathbf{c}^{(t)}), \tag{7}$$

$$\mathbf{h}^{(t)} = \mathbf{o}^{(t)} \odot \phi(\mathbf{c}^{(t)}), \tag{8}$$

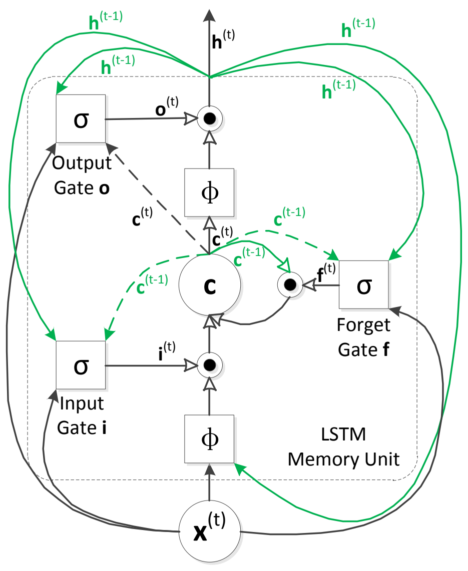


Figure 2: LSTM Memory Unit. The memory unit contains a cell **c** which is controlled by three gates. The green links show the signals at time $t-1$, while the black links show the current signals. The dashed links represent the weight matrices from beginning to end are diagonal. Moreover, the solid pointers mean there are weight matrices on the connections, and hollow pointers mean none. The current output signal, $\mathbf{h}^{(t)}$, will fed back to the next time $t+1$ via three gates, and is the input of the higher layer of the neural network as well.

where σ and ϕ are the logistic sigmoid function and hyperbolic tangent function respectively; $\mathbf{i}^{(t)}$, $\mathbf{f}^{(t)}$, $\mathbf{o}^{(t)}$ and $\mathbf{c}^{(t)}$ are respectively the input gate, forget gate, output gate, and memory cell activation vector at time step t, all of which have the same size as the hidden vector $\mathbf{h}^{(t)} \in \mathbb{R}^{H_2}$; the parameter matrices Ws with different subscripts are all square matrices; $\odot$ denotes the element-wise product of the vectors. Note that $\mathbf{W}_{ic}$, $\mathbf{W}_{fc}$ and $\mathbf{W}_{oc}$ are diagonal matrices.

3.3 LSTM Architectures for Chinese Word Segmentation

To fully utilize the LSTM, we propose four different structures of neural network to select the effective features via memory units. Figure 3 illustrates our proposed architectures.

LSTM-1 The LSTM-1 simply replace the hidden neurons in Eq. (1) with LSTM units (See Figure 3a).

The input of the LSTM unit is from a window of context characters. For each character, $c^{(t)}, (1 \leq t \leq n)$, the input of the LSTM unit $\mathbf{x}^{(t)}$,

$$\mathbf{x}^{(t)} = \mathbf{v}_c^{(t-k_1)} \oplus \cdots \oplus \mathbf{v}_c^{(t+k_2)}, \tag{9}$$

is concatenated from character embeddings of $c^{(t-k_1):(t+k_2)}$, where k_1 and k_2 represent the numbers of characters from left and right contexts respectively. The output of the LSTM unit is used in final inference function (Eq. (11)) after a linear transformation.

LSTM-2 The LSTM-2 can be created by stacking multiple LSTM hidden layers on top of each other, with the output sequence of one layer forming the input sequence for the next (See Figure 3b). Here we use two LSTM layers. Specifically, input of the upper LSTM layer takes $\mathbf{h}^{(t)}$ from the lower LSTM layer without any transformation. The input of the first layer is same to LSTM-1, and the output of the second layer is as same operation as LSTM-1.

LSTM-3 The LSTM-3 is a extension of LSTM-1, which adopts a local context of LSTM layer as input of the last layer (See Figure 3c). For each time step t, we concatenate the outputs of a window of the LSTM layer into a vector $\hat{\mathbf{h}}^{(t)}$,

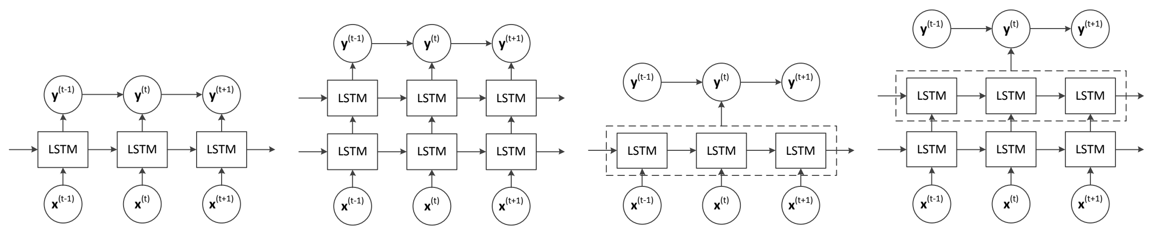
$$\hat{\mathbf{h}}^{(t)} = \mathbf{h}^{(t-m_1)} \oplus \cdots \oplus \mathbf{h}^{(t+m_2)}, \tag{10}$$

where m_1 and m_2 represent the lengths of time lags before and after current time step.Finally, $\hat{\mathbf{h}}^{(t)}$ is used in final inference function (Eq. (11)) after a linear transformation.

LSTM-4 The LSTM-4 (see Figure 3d) is a mixture of the LSTM-2 and LSTM-3, which consists of two LSTM layers. The output sequence of the lower LSTM layer forms the input sequence of the upper LSTM layer. The final layer adopts a local context of upper LSTM layer as input.

3.4 Inference at Sentence Level

To model the tag dependency, previous neural network models (Collobert et al., 2011; Zheng et al., 2013; Pei et al., 2014) introduced the transition score $\mathbf{A}_{ij}$ for measuring the probability of jumping from tag $i \in T$ to tag $j \in T$. For a input sentence $c^{(1:n)}$ with a tag sequence $y^{(1:n)}$, a sentence-level score is then given by the sum of tag transition scores and network tagging scores:



(a) LSTM-1 (b) LSTM-2 (c) LSTM-3 (d) LSTM-4

Figure 3: Our proposed LSTM architectures for Chinese word segmentation.

$$s(c^{(1:n)}, y^{(1:n)}, \theta) = \sum_{t=1}^{n} \left(\mathbf{A}_{y^{(t-1)} y^{(t)}} + \mathbf{y}^{(t)}_{y^{(t)}} \right), \tag{11}$$

where $\mathbf{y}^{(t)}_{y^{(t)}}$ indicates the score of tag $y^{(t)}$, and $\mathbf{y}^{(t)}$ is computed by the network as in Eq. (2). The parameter set of our model $\theta = \{\mathbf{M}, \mathbf{A}, \mathbf{W}_{ic}, \mathbf{W}_{fc}, \mathbf{W}_{oc}, \mathbf{W}_{ix}, \mathbf{W}_{fx}, \mathbf{W}_{ox}, \mathbf{W}_{ih}, \mathbf{W}_{fh}, \mathbf{W}_{oh}, \mathbf{W}_{cx}, \mathbf{W}_{ch}\}$.

4 Training

4.1 Max-Margin criterion

We use the Max-Margin criterion to train our model. Intuitively, the Max-Margin criterion provides an alternative to probabilistic, likelihood based estimation methods by concentrating directly on the robustness of the decision boundary of a model (Taskar et al., 2005). We use $Y(x_i)$ to denote the set of all possible tag sequences for a given sentence x_i and the correct tag sequence for x_i is y_i. The parameter set of our model is θ. We first define a structured margin loss $\Delta(y_i, \hat{y})$ for predicted tag sequence $\hat{y}$:

$$\Delta(y_i, \hat{y}) = \sum_{t}^{n} \eta \mathbf{1}\{y_i^{(t)} \neq \hat{y}^{(t)}\}, \tag{12}$$

where n is the length of sentence x_i and η is a discount parameter. The loss is proportional to the number of characters with incorrect tags in the proposed tag sequence. For a given training instance (x_i, y_i),the predicted tag sequence $\hat{y}_i \in Y(x_i)$ is the one with the highest score:

$$\hat{y}_i = \underset{y \in Y(x_i)}{\arg\max}\, s(x_i, y, \theta), \tag{13}$$

where the function $s(\cdot)$ is sentence-level score and defined in equation (11).

Given a set of training set $\mathcal{D}$, the regularized objective function is the loss function $J(\theta)$ including a l_2-norm term:

$$J(\theta) = \frac{1}{|\mathcal{D}|} \sum_{(x_i, y_i) \in \mathcal{D}} l_i(\theta) + \frac{\lambda}{2} \|\theta\|_2^2, \tag{14}$$

where $l_i(\theta) = \max(0, s(x_i, \hat{y}_i, \theta) + \Delta(y_i, \hat{y}_i) - s(x_i, y_i, \theta))$.

To minimize $J(\theta)$, we use a generalization of gradient descent called subgradient method (Ratliff et al., 2007) which computes a gradient-like direction.

Following (Socher et al., 2013), we also use the diagonal variant of AdaGrad (Duchi et al., 2011) with minibatchs to minimize the objective. The parameter update for the i-th parameter $\theta_{t,i}$ at time step t is as follows:

$$\theta_{t,i} = \theta_{t-1,i} - \frac{\alpha}{\sqrt{\sum_{\tau=1}^{t} g_{\tau,i}^2}} g_{t,i}, \tag{15}$$

where α is the initial learning rate and $g_\tau \in \mathbb{R}^{|\theta_i|}$ is the subgradient at time step τ for parameter θ_i. In addition, the process of back propagation is followd Hochreiter and Schmidhuber (1997).

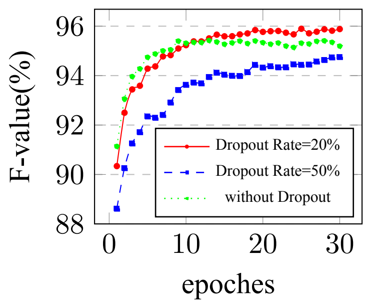
4.2 Dropout

Dropout is one of prevalent methods to avoid overfitting in neural networks (Srivastava et al., 2014). When dropping a unit out, we temporarily remove it from the network, along with all its incoming and outgoing connections. In the simplest case, each unit is omitted with a fixed probability p independent of other units, namely dropout rate, where p is also chosen on development set.

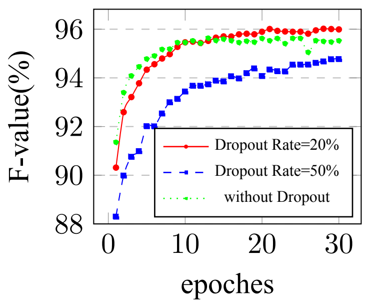
5 Experiments

5.1 Datasets

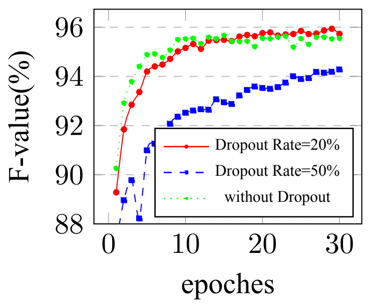
We use three popular datasets, PKU, MSRA and CTB6, to evaluate our model. The PKU



(a) LSTM-1(2,2)



(b) LSTM-1 (1,2)



(c) LSTM-1(0,2)

Figure 4: Performances of LSTM-1 with the different context lengths and dropout rates on PKU development set.

Context length	$(k_1, k_2) = (0, 2)$
Character embedding size	$d = 100$
Hidden unit number	$H_2 = 150$
Initial learning rate	$\alpha = 0.2$
Margin loss discount	$\eta = 0.2$
Regularization	$\lambda = 10^{-4}$
Dropout rate on input layer	$p = 0.2$

Table 1: Settings of the hyper-parameters.

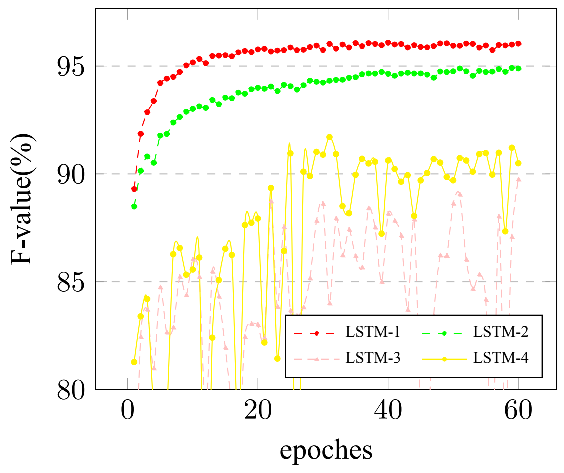


Figure 5: Performances of LSTM-1 (0,2) with 20% dropout on PKU development set.

and MSRA data are provided by the second International Chinese Word Segmentation Bakeoff (Emerson, 2005), and CTB6 is from Chinese TreeBank 6.0 (LDC2007T36) (Xue et al., 2005), which is a segmented, part-of-speech tagged and fully bracketed corpus in the constituency formalism. These datasets are commonly used by previous state-of-the-art models and neural network models. In addition, we use the first 90% sentences of the training data as training set and the rest 10% sentences as development set for PKU and MSRA datasets. For CTB6 dataset, we divide the training, development and test sets according to (Yang and Xue, 2012)

All datasets are preprocessed by replacing the Chinese idioms and the continuous English characters and digits with a unique flag.

For evaluation, we use the standard bake-off scoring program to calculate precision, recall, F1-score and out-of-vocabulary (OOV) word recall.

5.2 Hyper-parameters

Hyper-parameters of neural model impact the performance of the algorithm significantly. According to experiment results, we choose the hyper-parameters of our model as showing in Figure 1. The minibatch size is set to 20. Generally, the number of hidden units has a limited impact on the performance as long as it is large enough. We found that 150 is a good trade-off between speed and model performance. The dimensionality of character embedding is set to 100 which achieved the best performance. All these hyper-parameters are chosen according to their average performances on three development sets.

For the context lengths (k_1, k_2) and dropout strategy, we give detailed analysis in next section.

5.3 Dropout and Context Length

We first investigate the different dropout strategies, including dropout at different layers and with different dropout rate p. As a result, we found that it is a good trade-off between speed and model performance to drop the input layer only with dropout rate $p_{input} = 0.2$. However, it does not show any significant improvement to dropout on hidden LSTM layers.

Context Length	Dropout rate=20%			Dropout rate=50%			without Dropout		
	P	R	F	P	R	F	P	R	F
LSTM-1 (2,2)	95.8	95.3	95.6	94.8	94.4	94.6	95.2	94.9	95.1
LSTM-1 (1,2)	95.7	95.3	95.5	94.8	94.4	94.6	95.4	94.9	95.2
LSTM-1 (0,2)	**95.8**	**95.5**	**95.7**	94.6	94.2	94.4	95.4	95.0	95.2

Table 2: Performances of LSTM-1 with the different context lengths and dropout rates on PKU test set.

models	Contextr Length = (0,2)		
	P	R	F
LSTM-1	**95.8**	**95.5**	**95.7**
LSTM-2	95.1	94.5	94.8
LSTM-3	89.1	90.4	89.8
LSTM-4	92.1	91.7	91.9

Table 3: Performance on our four proposed models on PKU test set.

Due to space constraints, we just give the performances of LSTM-1 model on PKU dataset with different context lengths (k_1, k_2) and dropout rates in Figure 4 and Table 2. From Figure 4, we can see that 20% dropout converges slightly slower than the one without dropout, but avoids overfitting. 50% or higher dropout rate seems to be underfitting since its training error is also high.

Table 2 shows that the LSTM-1 model performs consistently well with the different context length, but the LSTM-1 model with short context length saves computational resource, and gets more efficiency. At the meanwhile, the LSTM-1 model with context length (0,2) can receive the same or better performance than that with context length (2,2), which shows that the LSTM model can well model the pervious information, and it is more robust for its insensitivity of window size variation.

We employ context length (0,2) with the 20% dropout rate in the following experiments to balance the tradeoff between accuracy and efficiency.

5.4 Model Selection

We also evaluate the our four proposed models with the hyper-parameter settings in Table 1. For LSTM-3 and LSTM-4 models, the context window length of top LSTM layer is set to (2,0). For LSTM-2 and LSTM-4,the number of upper hidden LSTM layer is set to 100. We use PKU dataset to select the best model. Figure 5 shows the results of the four models on PKU development set from first epoch to 60-th epoch. We see that the LSTM-1 is the fastest one to converge and achieves the best performance. The LSTM-2 (two LSTM layers) get worse, which shows the performance seems not to benefit from deep model. The LSTM-3 and LSTM-4 models do not converge, which could be caused by the complexity of models.

The results on PKU test set are also shown in Table 3, which again show that the LSTM-1 achieves the best performance. Therefore, in the rest of the paper we will give more analysis based on the LSTM-1 with hyper-parameter settings as showing in Table 1.

5.5 Experiment Results

In this section, we give comparisons of the LSTM-1 with pervious neural models and state-of-the-art methods on the PKU, MSRA and CTB6 datasets.

We first compare our model with two neural models (Zheng et al., 2013; Pei et al., 2014) on Chinese word segmentation task with random initialized character embeddings. As showing in Table 4, the performance is boosted significantly by utilizing LSTM unit. And more notably, our window size of the context characters is set to (0,2), while the size of the other models is (2,2).

Previous works found that the performance can be improved by pre-training the character embeddings on large unlabeled data. We use word2vec [1] (Mikolov et al., 2013a) toolkit to pre-train the character embeddings on the Chinese Wikipedia corpus. The obtained embeddings are used to initialize the character lookup table instead of random initialization. Inspired by (Pei et al., 2014), we also utilize bigram character embeddings which is simply initialized as the average of embeddings of two consecutive characters.

Table 5 shows the performances with additional pre-trained and bigram character embeddings. Again, the performances boost significantly as a result. Moreover, when we use bigram embeddings only, which means we do close test without pre-training the embeddings on other extra corpus, our model still perform competitively compared

[1] http://code.google.com/p/word2vec/

models	PKU			MSRA			CTB6		
	P	R	F	P	R	F	P	R	F
(Zheng et al., 2013)	92.8	92.0	92.4	92.9	93.6	93.3	94.0*	93.1*	93.6*
(Pei et al., 2014)	93.7	93.4	93.5	94.6	94.2	94.4	94.4*	93.4*	93.9*
LSTM	**95.8**	**95.5**	**95.7**	**96.7**	**96.2**	**96.4**	**95.0**	**94.8**	**94.9**

Table 4: Performances on three test sets with random initialized character embeddings. The results with * symbol are from our implementations of their methods.

models	PKU			MSRA			CTB6		
	P	R	F	P	R	F	P	R	F
+Pre-train									
(Zheng et al., 2013)	93.5	92.2	92.8	94.2	93.7	93.9	93.9*	93.4*	93.7*
(Pei et al., 2014)	94.4	93.6	94.0	95.2	94.6	94.9	94.2*	93.7*	94.0*
LSTM	96.3	95.6	96.0	96.7	96.5	96.6	95.9	95.5	95.7
+bigram									
LSTM	96.3	95.9	96.1	97.1	97.1	97.1	95.6	95.3	95.5
+Pre-train+bigram									
(Pei et al., 2014)	-	-	95.2	-	-	97.2	-	-	-
LSTM	**96.6**	**96.4**	**96.5**	**97.5**	**97.3**	**97.4**	**96.2**	**95.8**	**96.0**

Table 5: Performances on three test sets with pre-trained and bigram character embeddings. The results with * symbol are from our implementations of their methods.

Models	PKU	MSRA	CTB6
(Tseng et al., 2005)	95.0	96.4	-
(Zhang and Clark, 2007)	95.1	97.2	-
(Sun and Xu, 2011)	-	-	95.7
(Zhang et al., 2013)	96.1	97.4	-
This work	**96.5**	**97.4**	**96.0**

Table 6: Comparison of our model with state-of-the-art methods on three test sets.

with previous neural models with pre-trained embedding and bigram embeddings.

Table 6 lists the performances of our model as well as previous state-of-the-art systems. (Zhang and Clark, 2007) is a word-based segmentation algorithm, which exploit features of complete words, while the rest of the list are character-based word segmenters, whose features are mostly extracted from a window of characters. Moreover, some systems (such as Sun and Xu (2011) and Zhang et al. (2013)) also exploit kinds of extra information such as unlabeled data or other knowledge. Despite our model only uses simple bigram features, it outperforms previous state-of-the-art models which use more complex features.

Since that we do not focus on the speed of the algorithm in this paper, we do not optimize the speed a lot. On PKU dataset, it takes about 3 days to train the model (last row of Table 5) using CPU (Intel(R) Xeon(R) CPU E5-2665 @ 2.40GHz) only. All implementation is based on Python.

6 Related Work

Chinese word segmentation has been studied with considerable efforts in the NLP community. The most popular word segmentation methods is based on sequence labeling (Xue, 2003). Recently, researchers have tended to explore neural network based approaches (Collobert et al., 2011) to reduce efforts of feature engineering (Zheng et al., 2013; Pei et al., 2014; Qi et al., 2014; Chen et al., 2015). The features of all these methods are extracted from a local context and neglect the long distance information. However, previous information is also crucial for word segmentation. Our model adopts the LSTM to keep the previous important information in memory and avoids the limitation of ambiguity caused by limit of the size of context window.

7 Conclusion

In this paper, we use LSTM to explicitly model the previous information for Chinese word segmentation, which can well model the potential long-distance features. Though our model use smaller context window size (0,2), it still outperforms the previous neural models with context window size (2,2). Besides, our model can also be easily generalized and applied to other sequence labeling tasks.

Although our model achieves state-of-the-art performance, it only makes use of previous context. The future context is also useful for Chinese word segmentation. In future work, we would like to adopt the bidirectional recurrent neural network (Schuster and Paliwal, 1997) to process the sequence in both directions.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. This work was partially funded by the National Natural Science Foundation of China (61472088, 61473092), National High Technology Research and Development Program of China (2015AA015408), Shanghai Science and Technology Development Funds (14ZR1403200).

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155.

A.L. Berger, V.J. Della Pietra, and S.A. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, and Xuanjing Huang. 2015. Gated recursive neural network for Chinese word segmentation. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of ICML*.

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

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

T. Emerson. 2005. The second international Chinese word segmentation bakeoff. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, pages 123–133. Jeju Island, Korea.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*.

PengFei Liu, Xipeng Qiu, Xinchi Chen, Shiyu Wu, and Xuanjing Huang. 2015. Multi-timescale long short-term memory neural network for modelling sentences and documents. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

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

Wenzhe Pei, Tao Ge, and Chang Baobao. 2014. Max-margin tensor neural network for chinese word segmentation. In *Proceedings of ACL*.

F. Peng, F. Feng, and A. McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. *Proceedings of the 20th international conference on Computational Linguistics*.

Yanjun Qi, Sujatha G Das, Ronan Collobert, and Jason Weston. 2014. Deep learning for character-based information extraction. In *Advances in Information Retrieval*, pages 668–674. Springer.

Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. 2007. (online) subgradient methods for structured prediction. In *Eleventh International Conference on Artificial Intelligence and Statistics (AIStats)*.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*, 45(11):2673–2681.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013. Parsing with compositional vector grammars. In *In Proceedings of the ACL conference*. Citeseer.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Weiwei Sun and Jia Xu. 2011. Enhancing Chinese word segmentation using unlabeled data. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 970–979.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005. A conditional random field word segmenter for sighan bakeoff 2005. In *Proceedings of the fourth SIGHAN workshop on Chinese language Processing*, volume 171.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.

Oriol Vinyals, Suman V Ravuri, and Daniel Povey. 2012. Revisiting recurrent neural networks for robust asr. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 4085–4088. IEEE.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The Penn Chinese TreeBank: Phrase structure annotation of a large corpus. *Natural language engineering*, 11(2):207–238.

N. Xue. 2003. Chinese word segmentation as character tagging. *Computational Linguistics and Chinese Language Processing*, 8(1):29–48.

Yaqin Yang and Nianwen Xue. 2012. Chinese comma disambiguation for discourse analysis. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 786–794. Association for Computational Linguistics.

Yue Zhang and Stephen Clark. 2007. Chinese segmentation with a word-based perceptron algorithm. In *ACL*.

Longkai Zhang, Houfeng Wang, Xu Sun, and Mairgup Mansur. 2013. Exploring representations from unlabeled data with co-training for Chinese word segmentation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for chinese word segmentation and pos tagging. In *EMNLP*, pages 647–657.