The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Neural Networks Incorporating Dictionaries for Chinese Word Segmentation

Qi Zhang, Xiaoyu Liu, Jinlan Fu
Shanghai Key Laboratory of Intelligent Information Processing
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai, P.R. China
{qz, liuxiaoyu16, fujl16}@fudan.edu.cn

Abstract

In recent years, deep neural networks have achieved significant success in Chinese word segmentation and many other natural language processing tasks. Most of these algorithms are end-to-end trainable systems and can effectively process and learn from large scale labeled datasets. However, these methods typically lack the capability of processing rare words and data whose domains are different from training data. Previous statistical methods have demonstrated that human knowledge can provide valuable information for handling rare cases and domain shifting problems. In this paper, we seek to address the problem of incorporating dictionaries into neural networks for the Chinese word segmentation task. Two different methods that extend the bi-directional long short-term memory neural network are proposed to perform the task. To evaluate the performance of the proposed methods, state-of-the-art supervised models based methods and domain adaptation approaches are compared with our methods on nine datasets from different domains. The experimental results demonstrate that the proposed methods can achieve better performance than other state-of-the-art neural network methods and domain adaptation approaches in most cases.

Introduction

Chinese word segmentation (CWS) is an important and essential pre-processing step for Chinese language processing tasks. In recent years, most methods have treated the task as a sequential labelling problem (Xue 2003; Zhang et al. 2003). For a given piece of text, labels are assigned to all of the characters in that text, indicating the position of a character in the word (Xue 2003) or representing the intervals between characters (Huang et al. 2007). Various supervised learning approaches have also been carefully studied to achieve the task, including maximum entropy (ME) (Berger, Pietra, and Pietra 1996), support vector machines (SVM) (Boser, Guyon, and Vapnik 1992), hidden Markov models (HMMs) (Eddy 1996), conditional random fields (CRFs) (Lafferty, McCallum, and Pereira 2001), and others.

Recently, along with the development of deep learning methods, some neural network models have also been successfully used for Chinese word segmentation tasks (Zheng, Chen, and Xu 2013; Chen et al. 2015b; Zhang, Zhang, and Fu 2016). Zheng et al. (2013) adopted the method proposed by Collobert et al. (2011) to perform Chinese word segmentation and POS tagging. Chen et al. (2015b) extended LSTM to explicitly model previously important information in memory cells to perform the task. Experimental results have demonstrated that the performance of these methods can compete with previous state-of-the-art systems. A word-based neural model for Chinese word segmentation has also been proposed to exploit not only character embeddings but also word embeddings pre-trained from large scale corpus (Zhang, Zhang, and Fu 2016).

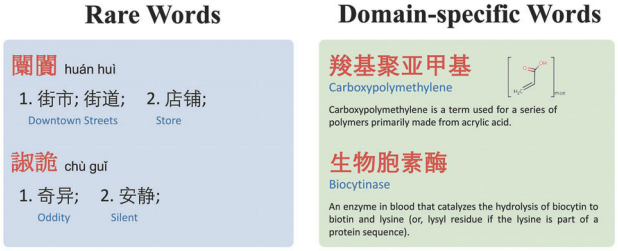

Figure 1: Examples of rare words and domain-specific words in dictionaries.

Despite the great success achieved by neural network based methods, some issues still have not been well solved. One significant drawback is that such methods rarely take the integration of knowledge into consideration. These networks usually employ an end-to-end approach and try to directly learn information from large scale labeled data. However there are also a huge number of cases that rarely occur, and neural networks usually cannot handle such cases well. Out-of-vocabulary (OOV) words are one of the most common errors of supervised CWS methods (Peng, Feng, and McCallum 2004). While, dictionaries contain both commonly used words and rare words. Figure 1 illustrates dictionary examples. Since these words are rarely used, they are normally not included in the labeled data. Hence, if we can incorporate a dictionary into a neural network, rare words and domain-specific words can be better processed.

Previous methods have shown that incorporating dictionaries can bring significant improvement to the Chinese

word segmentation task. Peng et al. (2004) proposed using CRF to perform the task. By analyzing the experimental results reported in the paper of Peng et al., we can see that a method that takes dictionary features into consideration can achieve much better performance than methods that do not take dictionary features into account. Qian and Liu (2012) introduced a method to jointly model Chinese word segmentation, POS tagging and parsing with features extracted from dictionaries. Liu et al. (2014) proposed a semi-supervised method that uses CRF with domain specific dictionaries for cross-domain Chinese word segmentation .

Although these approaches have demonstrated the usefulness of dictionaries on CWS task, existing neural network based CWS methods usually focused on constructing more complex network architecture and did not carefully study this problem. Several studies have examined related issues. Yang et al. (2017) exploited richer sources of external information, including punctuation, automatic segmentation and POS, to pretrain character and word embeddings to improve performance. Zhang et al. (2017) studied the problem of integrating prior knowledge for a neural machine translation task. They proposed a posterior regularization method to incorporate a phrase table, length ration, bilingual dictionary and coverage penalty to perform the machine learning task. Hence, how to incorporate dictionaries into neural networks for CWS should be investigated.

In this paper, we propose a novel method to achieve the task. We extend bi-directional long short-term memory and conditional random field (Bi-LSTM-CRF) (Huang, Xu, and Yu 2015) to model the CWS task as a character level sequence labeling problem. To integrate dictionaries, we define several templates to construct feature vectors for each character based on dictionaries and contexts. Then, two different methods are introduced to integrate feature vectors with character embeddings to perform the task. Three simplified Chinese and two traditional Chinese corpora are used for evaluation. Besides training and testing the proposed methods on the same domain, we also show that the proposed methods can achieve significantly better performance on the domain adaptation task. When applying the model on different domains, we only need to add extra domain specific dictionaries. The other learned parameters can remain unchanged with no need for retraining.

The contributions of this paper can be summarized as follows.

- We studied the problem of integrating dictionaries into neural networks based methods for the Chinese word segmentation task.
- We proposed two methods to integrate information extracted from dictionaries into neural network based methods for a CWS task. The methods can solve the problem caused by rare words and achieve significant improvements in the cross-domain CWS task.
- Extensive experiments on nine corpora are conducted to compare the proposed methods with state-of-the-art methods.

Bi-LSTM-CRF Model for Chinese Word Segmentation

The Chinese word segmentation task is usually regarded as a character-based sequence labeling task. Given a sentence $\mathbf{x} = (x_1, x_2, ..., x_n)$, each character x_i in the sentence $\mathbf{x}$ will be labeled as one of $T = \{B, M, E, S\}$, indicating the character is in the beginning, middle, end of a word, or the character is merely a single-character word. In this section, we will give a brief description of the general Bi-LSTM-CRF architecture for CWS.

Similar to other methods using neural networks, the first step of the Bi-LSTM-CRF based CWS method is to represent characters in distributed vectors. In this work, we use $\mathbf{e}_{x_i} \in \mathbb{R}^{d_e}$ to represent the embedding of character x_i.

Bi-LSTM Layer

The long short-term memory network (LSTM) (Hochreiter and Schmidhuber 1997) is a variant of the recurrent neural network (RNN). Although, in theory, the RNN can process any long-distance dependencies; in practice, it fails to do so as a result of gradient vanishing/exploding problems. The LSTM provides a solution by introducing gate mechanism and memory cell. Formally, the input gate $\mathbf{i}$, forget gate $\mathbf{f}$, memory cell $\mathbf{c}$, and output gate $\mathbf{o}$ could be defined as:

$$\begin{aligned}
\mathbf{i}_i &= \sigma(\mathbf{W}_{\mathrm{i}}\mathbf{h}_{i-1} + \mathbf{U}_{\mathrm{i}}\mathbf{e}_{x_i} + \mathbf{b}_{\mathrm{i}}) \\
\mathbf{f}_i &= \sigma(\mathbf{W}_{\mathrm{f}}\mathbf{h}_{i-1} + \mathbf{U}_{\mathrm{f}}\mathbf{e}_{x_i} + \mathbf{b}_{\mathrm{f}}) \\
\widetilde{\mathbf{c}}_i &= \tanh(\mathbf{W}_{\widetilde{\mathrm{c}}}\mathbf{h}_{i-1} + \mathbf{U}_{\widetilde{\mathrm{c}}}\mathbf{e}_{x_i} + \mathbf{b}_{\widetilde{\mathrm{c}}}) \\
\mathbf{c}_i &= \mathbf{f}_i \odot \mathbf{c}_{i-1} + \mathbf{i}_i \odot \widetilde{\mathbf{c}}_i \\
\mathbf{o}_i &= \sigma(\mathbf{W}_{\mathrm{o}}\mathbf{h}_{i-1} + \mathbf{U}_{\mathrm{o}}\mathbf{e}_{x_i} + \mathbf{b}_{\mathrm{o}}) \\
\mathbf{h}_i &= \mathbf{o}_i \odot \tanh(\mathbf{c}_i),
\end{aligned}$$

where σ and $\odot$ are the element-wise sigmoid function and element-wise product, respectively. $\mathbf{W}_{\mathrm{g}} \in \mathbb{R}^{4d_h \times d_h}$, $\mathbf{U}_{\mathrm{g}} \in \mathbb{R}^{4d_h \times d_e}$, and $\mathbf{b}_{\mathrm{g}} \in \mathbb{R}^{4d_h}$ are trainable parameters.

LSTM's hidden state $\mathbf{h}_i$ just takes information only from past, not considering future information. In order to incorporate information from both past and future, an elegant solution is to use bi-directional LSTM (Bi-LSTM) (Graves and Schmidhuber 2005). Specially, the hidden state of Bi-LSTM could be defined as:

$$\mathbf{h}_i = \overrightarrow{\mathbf{h}}_i \oplus \overleftarrow{\mathbf{h}}_i,$$

where $\overrightarrow{\mathbf{h}}_i$ and $\overleftarrow{\mathbf{h}}_i$ are the hidden states at position i of the forward and backward LSTM, respectively. $\oplus$ represents the concat operation. This representation has empirically proven to be efficient in capturing both past and future information.

CRF Layer

For the character-based Chinese word segmentation task, it is beneficial to consider the dependencies of adjacent labels. For example, a B (begin) label should be followed by a M (middle) label or E (end) label, and a M label cannot be followed by a B label or S (single) label. Therefore, instead of making tagging decisions using $\mathbf{h}_i$ independently, we employ a conditional random field (CRF) to model the label sequence jointly.

Formally, for a given sentence $\mathbf{x} = (x_1, x_2, ..., x_n)$ with a predicted tag sequence $\mathbf{y} = (y_1, y_2, ..., y_n)$, its prediction score can be defined as:

$$s(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} (\mathbf{A}_{y_{i-1} y_i} + \mathbf{P}_{i, y_i}),$$

where $\mathbf{A}$ is a transition score matrix and $\mathbf{A}_{i,j}$ measures the score of jumping from tag i to j. $\mathbf{P}_{i,y_i}$ represents the score of the y_i-th tag of the x_i. In particular, $\mathbf{P}_i$ could be precisely defined as:

$$\mathbf{P}_i = \mathbf{W}_s \mathbf{h}_i + \mathbf{b}_s$$

where $\mathbf{h}_i$ is the hidden state of Bi-LSTM at position i. $\mathbf{W}_s \in \mathbb{R}^{|T| \times d_h}$ and $\mathbf{b}_s \in \mathbb{R}^{|T|}$ are trainable parameters.

In the CRF layer, the probability of the sentence $\mathbf{x}$ being tagged for sequence $\mathbf{y}$ could be computed as:

$$p(\mathbf{y}|\mathbf{x}) = \frac{e^{s(\mathbf{x}, \mathbf{y})}}{\sum_{\widetilde{\mathbf{y}} \in \mathbf{Y}_\mathbf{x}} e^{s(\mathbf{x}, \widetilde{\mathbf{y}})}}.$$

where $\mathbf{Y}_\mathbf{x}$ represents all possible tag sequences $\widetilde{\mathbf{y}}$ for given sentence $\mathbf{x}$. For training, we can use the maximum likelihood estimation to maximize the log-probability $\log(p(\hat{\mathbf{y}}|\mathbf{x}))$ of the ground truth tag sequence $\hat{\mathbf{y}}$. While decoding, our prediction will be the tag sequence $\mathbf{y}^*$ with the highest score given by the following:

$$\mathbf{y}^* = \underset{\widetilde{\mathbf{y}} \in \mathbf{Y}_\mathbf{x}}{\operatorname{argmax}}\, s(\mathbf{x}, \widetilde{\mathbf{y}}).$$

We can use the Viterbi algorithm, a dynamic programming algorithm, to solve the efficiency problem in the training and decoding process.

Incorporating Dictionaries for Chinese Word Segmentation

From the brief description given above, we can observe that the Bi-LSTM-CRF model can learn information from large-scale labeled data. However, it cannot process rare words and domain-specific words very well. Hence, in this work, inspired by the success of integrating dictionaries into the CRF models for CWS (Low, Ng, and Guo 2005; Chang, Galley, and Manning 2008; Liu et al. 2014), we consider integrating dictionaries into neural networks based models.

For a given sentence $\mathbf{x} = (x_1, x_2, ..., x_n)$, we first construct feature vector $\mathbf{t}_i$ for each character x_i based on dictionary D and the context. The feature vector $\mathbf{t}_i$ represents whether character segments that consist of character x_i and its surroundings are words or not. After that, we propose two methods to integrate the feature vector $\mathbf{t}_i$ into the Bi-LSTM-CRF model. We will detail the feature vector construction and proposed methods in the following section.

Feature Vector Construction

As described above, given a sentence $\mathbf{x}$ and an external dictionary D, we first construct text segments based on the context of x_i using the pre-defined feature templates. The feature templates used in our work are listed in Table 1. For a text segment that appears in a feature template, we can generate a binary value to indicate whether the text segment is a word in D or not. $\mathbf{t}_{i_k}$ represents the value of the output corresponding to the k-th template for x_i. Finally, we generate an 8-dimensional vector containing word boundary information extracted from the dictionary D. Figure 2 illustrates an example of feature vector construction. Text segments are generated based on the feature templates shown in Table 1. Then, using the given dictionary D, we can obtain feature vector $\mathbf{t}_i$ for each character x_i.

Type	Template
2-gram	$x_{i-1}x_i$, x_ix_{i+1}
3-gram	$x_{i-2}x_{i-1}x_i$, $x_ix_{i+1}x_{i+2}$
4-gram	$x_{i-3}x_{i-2}x_{i-1}x_i$, $x_ix_{i+1}x_{i+2}x_{i+3}$
5-gram	$x_{i-4}x_{i-3}, ..., x_i$, $x_ix_{i+1}, ..., x_{i+4}$

Table 1: Feature templates for the i-th character, which are used to generate feature vector $\mathbf{t}_i$.

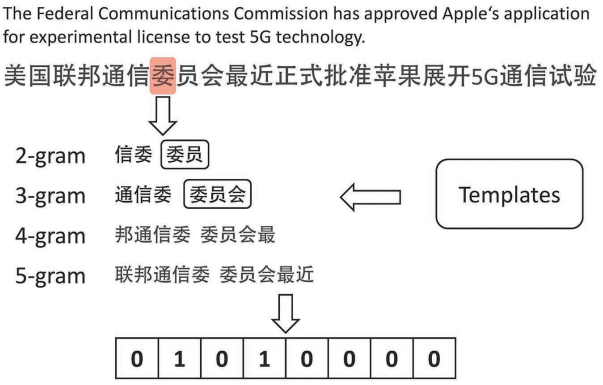

Figure 2: Example of feature vector construction. The character with the red shadow is the character x_i. The character segments with rounded rectangle are the words in the dictionary D.

To some degree, the feature vector can represent the candidate labels of a character based on the given dictionary. The values in the feature vector are dependent on the context and dictionary. They are not impacted by other sentences or statistical information. Hence, feature vectors can provide much information quite different from statistical-based methods. One should also note that there are other more complex ways to construct feature vector $\mathbf{t}_i$. For example, we can construct more feature templates and use more sophisticated features. However, the simple way is encouraging, and the focus of our work is not feature engineering. Therefore, we do not explore these alternate ways, leaving them for future work.

Model-I

Through introducing the feature vector construction step, given a sentence $\mathbf{x}$, we obtain both character embedding $\mathbf{e}_{x_i}$ and feature vector $\mathbf{t}_i$ for each character x_i. As described

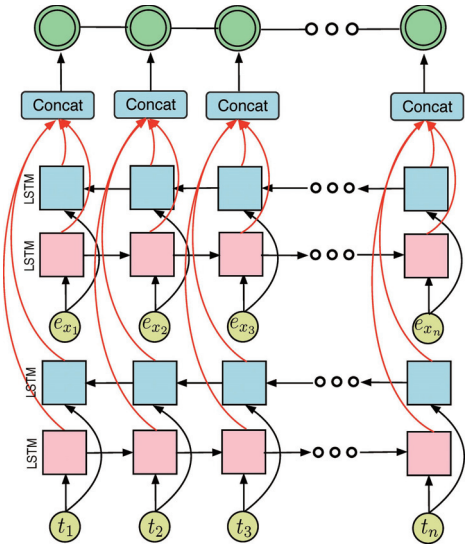

Figure 3: Main architecture of Model-I. The $\mathbf{e}_x$ and $\mathbf{t}$ represent character embeddings and feature vectors. The two parallel Bi-LSTMs are used to extract context information and potential word boundary information.

above, the basic Bi-LSTM-CRF-based model takes only the $\mathbf{e}_{x_i}$ as inputs. Because feature vectors could provide valuable information about different aspects, we propose to integrate it with the original Bi-LSMT-CRF model using another Bi-LSTM network.

The general architecture of the proposed model is illustrated in Figure 3. The two parallel Bi-LSTM can extract context information and potential word boundary information, respectively. For sentence $\mathbf{x}$, the hidden states of the two parallel Bi-LSTM at position i can be defined as:

$$\begin{aligned}\mathbf{h}_i^{\mathrm{x}} &= \text{Bi-LSTM}(\overrightarrow{\mathbf{h}}_{i-1}^{\mathrm{x}}, \overleftarrow{\mathbf{h}}_{i+1}^{\mathrm{x}}, \mathbf{e}_{x_i}) \\ \mathbf{h}_i^{\mathrm{t}} &= \text{Bi-LSTM}(\overrightarrow{\mathbf{h}}_{i-1}^{\mathrm{t}}, \overleftarrow{\mathbf{h}}_{i+1}^{\mathrm{t}}, \mathbf{t}_i),\end{aligned}$$

where $\mathbf{e}_{x_i}$ denotes the embedding vector of x_i, and $\mathbf{t}_i$ represents the feature vector generated through an external dictionary and the context of x_i. Note that in our formulation, the two parallel Bi-LSTM are independent, without any shared parameters. Then we combine the two hidden states as the inputs of CRF layer. Specifically, we simply adopt the concat operation to combine $\mathbf{h}_i^{\mathrm{x}}$ and $\mathbf{h}_i^{\mathrm{t}}$:

$$\mathbf{h}_i = \mathbf{h}_i^{\mathrm{x}} \oplus \mathbf{h}_i^{\mathrm{t}}$$

The other part of this model uses the same operation as the basic Bi-LSTM-CRF model.

Model-II

As described above, feature vectors represent the different boundary candidates. Under different boundary candidates, the weights for modeling x_i should be different. However, traditional LSTM have the weight-sharing constraints. Inspired by Ha et al. (2016), we consider the use of a hypernetwork to incorporate the information extracted from dictionaries. Specifically, we use $\mathbf{t}_i$ as the inputs of a Bi-LSTM

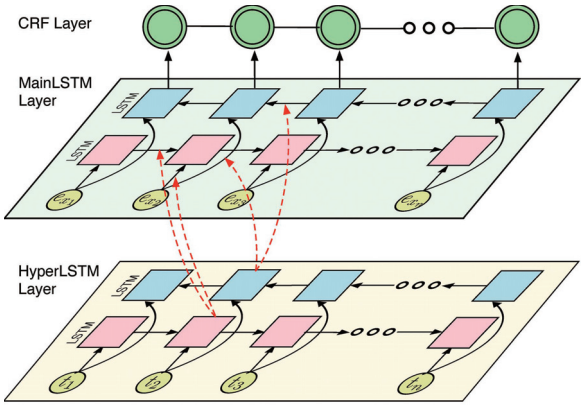

Figure 4: Main architecture of Model-II. We use a HyperLSTM to dynamically generate the weights for the MainLSTM. The HyperLSTM takes feature vectors $\mathbf{t}$ as inputs, while the MainLSTM takes embedding vectors $\mathbf{e}_x$ as inputs.

to generate the weights for modeling x_i. Figure 4 illustrates the general architecture of the proposed model. The bottom Bi-LSTM, which is called the HyperLSTM Layer, is used to generate the weights for the top Bi-LSTM. The top Bi-LSTM, which is called the MainLSTM layer, is used to model the character sequence $\{x_1, ..., x_n\}$.

The HyperLSTM cell has its own input sequence and hidden units, and these hidden units will be used to generate the weights for the MainLSTM. It provides an elegant way to relax the hard weight-sharing constraints of the traditional LSTM. This operation allows the weights of the MainLSTM to change under different boundary candidates, which might provide better results.

In contrast to Ha et al. (2016), the input to the HyperLSTM in our model is only the feature vector $\mathbf{t}_i$, and does not contain the hidden state of the MainLSTM $\mathbf{h}_{i-1}$ and the embedding vector $\mathbf{e}_{x_i}$. The weights for each of the four $\{\mathbf{i}, \mathbf{f}, \widetilde{\mathbf{c}}, \mathbf{o}\}$ gates of the MainLSTM will be generated by a set of embedding vectors $\mathbf{z}_x, \mathbf{z}_h, \mathbf{z}_b$ unique to each gate. These embedding vectors are linear projection of the hidden states of the HyperLSTM, and their dimension is typically smaller than the hidden unit number $d_{h^{\mathrm{t}}}$ of the HyperLSTM. For brevity and clarity, we use $\mathbf{g}$ to represent one of $\{\mathbf{i}, \mathbf{f}, \mathbf{o}, \widetilde{\mathbf{c}}\}$ instead of writing four sets of identical equations:

$$\begin{aligned}\mathbf{h}_i^{\mathrm{t}} &= \text{LSTM}(\mathbf{h}_{i-1}^{\mathrm{t}}, \mathbf{t}_i) \\ \mathbf{z}_{\mathrm{h}}^{\mathrm{g}} &= \mathbf{W}_{\mathrm{hh^t}}^{\mathrm{g}} \mathbf{h}_i^{\mathrm{t}} + \mathbf{b}_{\mathrm{hh^t}}^{\mathrm{g}} \\ \mathbf{z}_{\mathrm{x}}^{\mathrm{g}} &= \mathbf{W}_{\mathrm{xh^t}}^{\mathrm{g}} \mathbf{h}_i^{\mathrm{t}} + \mathbf{b}_{\mathrm{xh^t}}^{\mathrm{g}} \\ \mathbf{z}_{\mathrm{b}}^{\mathrm{g}} &= \mathbf{W}_{\mathrm{bh^t}}^{\mathrm{g}} \mathbf{h}_i^{\mathrm{t}}\end{aligned}$$

where $\mathbf{W}_{\mathrm{hh^t}}^{\mathrm{g}}, \mathbf{W}_{\mathrm{xh^t}}^{\mathrm{g}}, \mathbf{W}_{\mathrm{bh^t}}^{\mathrm{g}} \in \mathbb{R}^{d_z \times d_{h^t}}$, and $\mathbf{b}_{\mathrm{hh^t}}^{\mathrm{g}}, \mathbf{b}_{\mathrm{xh^t}}^{\mathrm{g}} \in \mathbb{R}^{d_z}$ are trainable parameters of linear projection layer. The $\mathbf{h}^{\mathrm{t}}$ is the hidden state of HyperLSTM.

Then, we use the embedding vectors $\mathbf{z}$ to generate the weights of the MainLSTM. Considering the memory efficiency, we use weight scaling vectors $\mathbf{d}$ to modify the rows of weight matrices instead of generating weight matrices di-

Datasets		PKU	MSR	AS	CITYU	CTB6	Literature	Computer	Medicine	Finance
Training set	#sent	19.1K	86.9K	709.0K	53.0K	25.5K	PKU's training set			
	#word	1.11M	2.37M	5.45M	1.46M	0.70M				
	#char	1.83M	4.05M	8.37M	2.40M	1.16M				
Testing set	#sent	1.9K	4.0K	14.4K	1.5K	2.8K	0.7K	1.3K	1.3K	0.6K
	#word	0.10M	0.11M	0.12M	0.04M	0.08M	35.2K	35.3K	31.5K	33.0K
	#char	0.17M	0.18M	0.20M	0.07M	0.13M	50.3K	64.2K	52.2K	56.3K
OOV Rate		5.8%	2.6%	4.3%	7.2%	5.3%	7.0%	15.2%	11.0%	8.7%

Table 2: Statistics of the nine different datasets.

rectly. The gate $\mathbf{g} \in \{\mathbf{i}, \mathbf{g}, \mathbf{f}, \mathbf{o}\}$ of the MainLSTM can be computed:

$$\begin{aligned}
\mathbf{g}_i &= \mathbf{d}_{\mathrm{h}}^{\mathrm{g}} \odot \mathbf{W}_{\mathrm{g}} \mathbf{h}_{i-1}^{\mathrm{x}} + \mathbf{d}_{\mathrm{x}}^{\mathrm{g}} \odot \mathbf{U}_{\mathrm{g}} \mathbf{e}_{x_i} + \mathbf{b}^{\mathrm{g}} \\
\mathbf{d}_{\mathrm{h}}^{\mathrm{g}} &= \mathbf{W}_{\mathrm{hz}}^{\mathrm{g}} \mathbf{z}_{\mathrm{h}}^{\mathrm{g}} \\
\mathbf{d}_{\mathrm{x}}^{\mathrm{g}} &= \mathbf{W}_{\mathrm{xz}}^{\mathrm{g}} \mathbf{z}_{\mathrm{x}}^{\mathrm{g}} \\
\mathbf{b}^{\mathrm{g}} &= \mathbf{W}_{\mathrm{bz}}^{\mathrm{g}} \mathbf{z}_{\mathrm{b}}^{\mathrm{g}} + \mathbf{b}_0^{\mathrm{g}}
\end{aligned}$$

where $\mathbf{W}_{\mathrm{hz}}^{\mathrm{g}}, \mathbf{W}_{\mathrm{xz}}^{\mathrm{g}}, \mathbf{W}_{\mathrm{bz}}^{\mathrm{g}} \in \mathbb{R}^{d_{h^x} \times d_z}$ and $\mathbf{b}_0^{\mathrm{g}} \in \mathbb{R}^{d_{h^x}}$.

Considering the fact that the backward HyperLSTM and MainLSTM are the same as the forward ones, we will not describe them.

Experiments

Datasets

We evaluated our models on nine frequently used CWS datasets, including SIGHAN2005 (Emerson 2005), CTB6 (Xue et al. 2005), and SIGHAN2010 (Zhao and Liu 2010). Table 2 lists the statistics of the nine datasets. Among these datasets, PKU, MSR, AS, CITYU (from SIGHAN2005), and CTB6 (from Chinese TreeBank 6.0) have been commonly used by previous state-of-the-art models. Note that AS and CITYU are traditional Chinese, and we map them into simplified Chinese before segmentation. SIGHAN2010 is usually used to evaluate domain adaptation algorithms for CWS. It contains four different test sets from the literature, computer, medical, and financial fields, while its training set is the same as PKU.

We used the simplified Chinese dictionary[1] sourced from jieba (a popular open source project for CWS) as the external dictionary in our work. In particular, for AS and CITYU, we added an extra traditional Chinese dictionary[2], taken from the Taiwan version of jieba. All datasets are preprocessed by replacing the Chinese idioms[3], continuous English characters and digits with a unique token. For evaluation, we use the standard bake-off scoring program to calculate F1 scores.

Experimental Configurations

Hyper-parameters may influence the performance of a neural network model. We adopted the hyper-parameters as shown in Table 3 for most of the datasets. The hidden unit number 2 d_{h^t} represents the hidden unit number of the LSTM, which takes feature vectors $\mathbf{t}$ as inputs in our two proposed models. The hyper embedding size d_z is the embedding vectors dimension of HyperLSTM in our model-II. In particular, we adjusted the hidden unit number d_{h^x} of PKU to 64, hidden unit number 2 d_{h^t} of SIGHAN2010 corpus to 160, and initial learning rate of MSR and AS to 0.001, while the remaining hyper-parameters remained unchanged.

The character embeddings used in our work were pre-trained using the word2vec (Mikolov et al. 2013) toolkit on the Chinese Wikipedia corpus and fine tuned in the training process. Following previous work (Pei, Ge, and Chang 2014; Chen et al. 2015b; 2017), we also used bigram character embeddings, which were initialized by averaging the embeddings of two contiguous characters.

Character embedding size	$d_e = 100$
Initial learning rate	$\alpha = 0.01$
Dropout rate	$p = 0.2$
Batch size	$b = 128$
Gradient clipping	$c = 5$
Hidden unit number	$d_{h^x} = 128$
Hidden unit number 2	$d_{h^t} = 128$
Hyper embedding size	$d_z = 16$

Table 3: Hyper-parameters configuration

Results

In this section, we first give the experimental results for five commonly used datasets for CWS. Then, we will discuss the performance of our models on the cross-domain CWS.

In-domain evaluation Table 4 lists the main results for five commonly used datasets, where the training and testing sets are in the same domain. For PKU, MSR, AS, and CITYU, only the training and testing sets were provided, and we used the first 90% of the sentences of the training set for training and the remaining 10% of the sentences as a development set. For CTB6, we followed Zhang et al. (2014) in constructing training, development, and testing sets.

In the first block, we give the performance of the latest neural network models without external resources, with the exception of pre-trained character embeddings. Cai et al. (2017) designed a word-based greedy search algorithm to improve the performance. Specially, Zhang et al. (2016) (hybrid) used a hybrid neural model, which integrated manual discrete features into their word-based neural network.

[1] https://github.com/fxsjy/jieba/tree/master/jieba/dict.txt.

[2] https://github.com/ldkrsi/jieba-zh_TW/blob/master/jieba.

[3] This idiom dictionary comes from Cai and Zhao (2016), and it is released at https://github.com/jcyk/CWS/tree/master/data/idioms

Models	PKU	MSR	AS	CITYU	CTB6
Cai and Zhao (2016)	95.5	96.5	-	-	-
Zhang et al. (2016) neural	95.1	97.0	-	-	95.0
Zhang et al. (2016) hybrid	95.7	97.7	-	-	96.0
Cai et al. (2017)	95.8	97.1	95.6	95.3	-
Chen et al. (2015a)*	96.1	96.2	-	-	95.8
Chen et al. (2015b)*	96.0	96.6	-	-	96.0
Cai and Zhao (2016)*	95.7	96.5	-	-	-
Chen et al. (2017)*	-	96.0	94.9	-	-
Cai et al. (2017)*	96.1	97.3	-	-	-
Yang et al. (2017)*	96.3	97.5	95.7	**96.9**	96.2
Bi-LSTM-CRF	95.1	97.0	95.3	95.3	95.6
Stacked Bi-LSTM-CRF	95.3	96.9	95.3	95.3	95.6
Model-I	96.2	97.6	95.6	96.0	96.1
Model-II	**96.5**	**97.8**	**95.9**	96.3	**96.4**

Table 4: Results on in-domain evaluation. There are four blocks. The first two blocks contains the latest neural network models, and the symbol * represents allowing the use of external resources. The last two blocks give the results of the baseline models and proposed models, respectively.

In the second block, we give the performance of the latest neural network models, allowing the use of external resources. Chen et al. (2015a; 2015b), Cai and Zhao (2016) and Cai et al. (2017) used an idiom dictionary[4] to replace Chinese idioms with a unique token. Chen et al. (2017) adopted an adversarial multi-criteria learning method to integrate shared knowledge from multiple heterogeneous segmentation criteria. Yang et al. (2017) exploited richer sources of external information, including punctuation, automatic segmentation, heterogeneous segmentation criteria, and POS data.

In the third block, we give the performance of two baseline models that don't use an external dictionary: Bi-LSTM-CRF and stacked Bi-LSTM-CRF. The stacked Bi-LSTM represents a two-layer Bi-LSTM. Based on the experimental results, we can see that merely increasing the depth of the Bi-LSTM cannot improve the performance.

In the last block, we give the performance of our two proposed models. We can note that our model-II achieves the best results on all of the datasets except for CITYU. This might be because that the CITYU source is from the Hong Kong corpora, and there are large differences between it and our dictionary. Simultaneously, compared with the two baseline models, the proposed models significantly improve the performance with the help of the information extracted from the external dictionary. This proves that incorporating dictionaries into neural network models can significantly boost the performance of the CWS. In particular, model-II performs better than model-I, possibly because model-II relaxed the hard weight-sharing constraints of the traditional LSTM and provided a more flexible way to extract features for each character x_i based on the dictionary and contexts.

[4]Their idiom dictionary was the same as the idiom dictionary used in our work. Note that we didn't use the original idiom dictionary for Chen et al. (2015a; 2015b), because it was neither publicly released nor specified the source until now. We give the results of re-running their code using the given idiom dictionary.

Models	Literature	Computer	Medicine	Finance
Jiang et al. (2013)	93.53	91.19	93.34	93.16
Liu et al. (2014)				
CRF+PA(Natural)	92.49	93.93	92.47	95.54
CRF+PA(Dict)	-	93.47	91.68	-
CRF+PA(Natural+Dict)	-	94.07	92.63	-
Bi-LSTM-CRF	93.05	93.20	91.85	95.20
Stacked Bi-LSTM-CRF	93.00	93.26	91.79	95.11
Model-I	92.61	92.32	91.18	94.64
Model-I + Domain dict	94.42	94.39	93.93	95.70
Model-II	92.87	92.65	91.27	94.95
Model-II + Domain dict	**94.76**	**94.70**	**94.18**	**96.06**

Table 5: Results on cross-domain evaluation. The first two blocks contain latest domain adaption models. In particular, PA, Natural, and Dict represent the partial annotation, natural annotation and dictionary. The last two blocks give the results of the baseline models and proposed models, and domain dict represents domain-specific dictionary.

Evaluation on cross-domain datasets We also compared our models with latest domain adaptation models for the cross-domain task, and the main results are listed in Table 5. For these datasets, no corresponding development set was provided. For a fair comparison, we did not select partial sentences from the testing set as a development set, but instead used the models trained on the training set for a fixed epoch for testing.

In particular, our external dictionary contained most domain-specific words of testing sets. In order to prove that our models could profit from domain-specific dictionaries for cross-domain tasks, we filtered partial domain-specific words appearing in the external dictionary as domain-specific dictionaries[5] and only used them during testing.

In the first two blocks, we give the experimental results of the latest domain adaptation models for CWS. Jiang et al. (2013) utilized linguistic knowledge in a large number of natural annotations to improve the performance of the cross-domain CWS. Liu et al (2014) exploited a CRF-based model, leveraging partial annotated data from dictionaries and natural annotation to help CWS on different domains.

In the last two blocks, we give the experimental results of the baseline models and our two proposed models. When not using domain-specific dictionaries, our proposed models perform the same or a little worse than the baseline Bi-LSTM-CRF models. This may be because we filtered partial domain-specific words of testing sets from the external dictionary, and the external dictionary cannot provide valuable information for domain-specific words. However, when these filtered domain-specific words were added during testing, our models achieved a great improvement and obtained the state-of-the-art performance. This proves that our models could profit from domain-specific dictionaries. In particular, when our models are applied on a different domain corpus, we only need to add an extra domain-specific dictionary rather than retrain the models. Therefore, we can easily apply our models to different domains for CWS.

[5]We regarded these out-of-vocabulary words appearing in the external dictionary as domain-specific words and randomly extracted some of them as the domain-specific dictionaries.

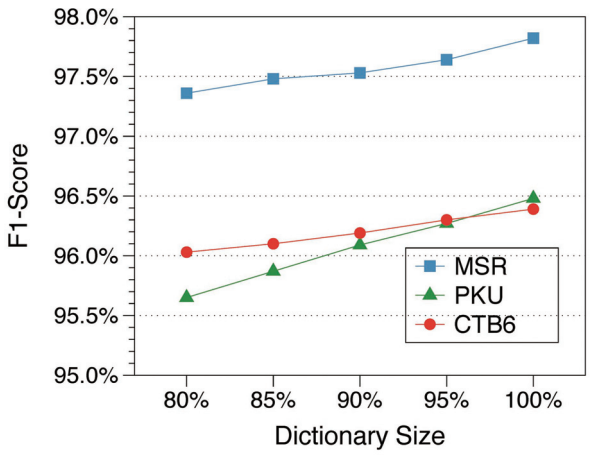

Figure 5: F1-scores with different sized dictionaries for Model-II. We randomly select different proportions of words from the original dictionary to generate new dictionaries.

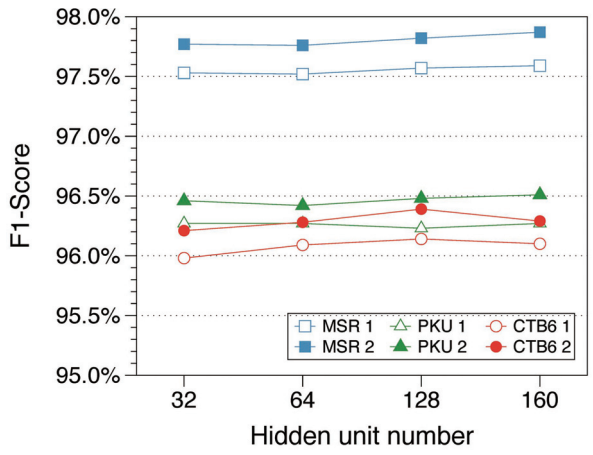

Figure 6: F1-scores with different hidden unit number d_{h^t}. The suffix numbers 1 and 2 represent the results of Model-I and Model-II, respectively.

Parameter analysis

In this section, we investigate the impact of the dictionary's size and hidden unit number d_{h^t}. To simplify the experiments, all of the experiments were only conducted on the PKU, MSR, and CTB6 corpus.

Dictionary size We first investigated the effect of the dictionary size. We randomly selected 80%, 85%, 90% and 95% of the words from the original dictionary to construct new dictionaries with different sizes. Figure 5 shows the F1-scores of our model-II with these dictionaries. From this figure, we can see the performance of the proposed model improves gradually with an increase in the dictionary size. Therefore, we could infer that we will get better results if we can obtain a dictionary containing more words.

Hidden unit number Then, we further investigate the influence of the hidden unit number d_{h^t} of the LSTM, which takes feature vectors $\mathbf{t}$ as the inputs in our proposed models. In our experiment, d_{h^t} was set 32, 64, 128, and 160, respectively. The results of our proposed models are shown in Figure 6. From this figure, we can see that the hidden unit number d_{h^t} has no discernible effect on the results of the proposed models, and our model-II performs better than model-I with different hidden unit number d_{h^t}.

Related work

Chinese word segmentation is always an active area in NLP tasks, and there have recently been many studies that have exploited various external resources to further improve the performance. These works mainly focused on statistical models and neural network models.

Liu et al. (2005) and Chang et al. (2008) both proposed that incorporating external dictionaries into statistical models can boost the performance. Zhao et al. (2010) systematically investigated multiple external resources, including an external dictionary, external name entity recognizer, and assistant word segmenter. Liu et al. (2014) exploited a CRF-based model, leveraging partial annotated data from dictionaries and natural annotation to help the CWS on different domains.

Chen et al. (2015a; 2015b) used an idiom dictionary to replace Chinese idioms with a unique token, and then adopted neural network models for CWS. Chen et al. (2017) proposed an adversarial multi-criteria learning method to integrate shared knowledge from multiple heterogeneous segmentation criteria. Yang et al. (2017) investigated the influences of various external resources, including punctuation, automatic segmentation, and POS, for neural word segmentation. They regarded each external resource as an auxiliary classification task, and used multi-task learning methods to pre-train the shared parameters used for the context modeling of Chinese characters.

Unlike the above models, we investigated the problem of incorporating dictionaries into neural network models.

Conclusion

In this study, we examined the problem of integrating dictionaries into neural network-based Chinese word segmentation methods. We proposed a method to extract information based on given dictionaries for a sentence. Then, two different models were introduced to use the information extracted from these dictionaries. Because dictionaries contain rare words and domain specific words, the models could process them better than previous methods. Experimental results showed that incorporating dictionaries could significantly enhance neural word segmentation. Our models achieved better results than state-of-the-art neural network models and domain adaptation models.

Acknowledgments

The research work is supported by the National Key Research and Development Program of China under Grant (No.

2017YFB1002104) and National Natural Science Foundation of China (No. 61532011, 61473092).

References

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

Boser, B. E.; Guyon, I. M.; and Vapnik, V. N. 1992. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*. ACM.

Cai, D., and Zhao, H. 2016. Neural word segmentation learning for chinese. In *ACL*.

Cai, D.; Zhao, H.; Zhang, Z.; Xin, Y.; Wu, Y.; and Huang, F. 2017. Fast and accurate neural word segmentation for chinese. In *ACL*.

Chang, P.-C.; Galley, M.; and Manning, C. D. 2008. Optimizing chinese word segmentation for machine translation performance. In *Proceedings of the third workshop on statistical machine translation*, 224–232.

Chen, X.; Qiu, X.; Zhu, C.; and Huang, X. 2015a. Gated recursive neural network for chinese word segmentation. In *ACL (1)*, 1744–1753.

Chen, X.; Qiu, X.; Zhu, C.; Liu, P.; and Huang, X. 2015b. Long short-term memory neural networks for chinese word segmentation. In *EMNLP*.

Chen, X.; Shi, Z.; Qiu, X.; and Huang, X. 2017. Adversarial multi-criteria learning for chinese word segmentation. In *ACL*.

Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; and Kuksa, P. 2011. Natural language processing (almost) from scratch. *JMLR*.

Eddy, S. R. 1996. Hidden markov models. *Current opinion in structural biology*.

Emerson, T. 2005. The second international chinese word segmentation bakeoff. In *Proceedings of the fourth SIGHAN workshop on Chinese language Processing*.

Graves, A., and Schmidhuber, J. 2005. Framewise phoneme classification with bidirectional lstm networks. In *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, volume 4, 2047–2052.

Ha, D.; Dai, A.; and Le, Q. V. 2016. Hypernetworks. *ICLR*.

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

Huang, C.-R.; Šimon, P.; Hsieh, S.-K.; and Prévot, L. 2007. Rethinking chinese word segmentation: tokenization, character classification, or wordbreak identification. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*.

Huang, Z.; Xu, W.; and Yu, K. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Jiang, W.; Sun, M.; Lü, Y.; Yang, Y.; and Liu, Q. 2013. Discriminative learning with natural annotations: Word segmentation as a case study. In *ACL*.

Lafferty, J.; McCallum, A.; and Pereira, F. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

Liu, Y.; Zhang, Y.; Che, W.; Liu, T.; and Wu, F. 2014. Domain adaptation for crf-based chinese word segmentation using free annotations. In *EMNLP*, 864–874.

Low, J. K.; Ng, H. T.; and Guo, W. 2005. A maximum entropy approach to chinese word segmentation. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, volume 1612164, 448–455.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Pei, W.; Ge, T.; and Chang, B. 2014. Max-margin tensor neural network for chinese word segmentation. In *ACL (1)*.

Peng, F.; Feng, F.; and McCallum, A. 2004. Chinese segmentation and new word detection using conditional random fields. In *Proceedings of the 20th international conference on Computational Linguistics*, 562.

Qian, X., and Liu, Y. 2012. Joint chinese word segmentation, pos tagging and parsing. In *EMNLP*, 501–511.

Xue, N.; Xia, F.; Chiou, F.-D.; and Palmer, M. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Natural language engineering* 11(2):207–238.

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

Yang, J.; Zhang, Y.; and Dong, F. 2017. Neural word segmentation with rich pretraining. In *ACL*.

Zhang, H.-P.; Yu, H.-K.; Xiong, D.-Y.; and Liu, Q. 2003. Hhmm-based chinese lexical analyzer ictclas. In *Proceedings of the second SIGHAN workshop on Chinese language processing-Volume 17*.

Zhang, M.; Zhang, Y.; Che, W.; and Liu, T. 2014. Character-level chinese dependency parsing. In *ACL (1)*, 1326–1336.

Zhang, J.; Liu, Y.; Luan, H.; Xu, J.; and Sun, M. 2017. Prior knowledge integration for neural machine translation using posterior regularization. In *ACL*.

Zhang, M.; Zhang, Y.; and Fu, G. 2016. Transition-based neural word segmentation. In *ACL*.

Zhao, H., and Liu, Q. 2010. The cips-sighan clp 2010 chinese word segmentation bakeoff. In *Proceedings of the First CPS-SIGHAN Joint Conference on Chinese Language Processing*, 199–209.

Zhao, H.; Huang, C.-N.; Li, M.; and Lu, B.-L. 2010. A unified character-based tagging framework for chinese word segmentation. *ACM Transactions on Asian Language Information Processing (TALIP)* 9(2):5.

Zheng, X.; Chen, H.; and Xu, T. 2013. Deep learning for chinese word segmentation and pos tagging. In *EMNLP*.