Multiple Character Embeddings for Chinese Word Segmentation

Jingkang Kang
Shanghai Jiao Tong University
Shanghai, China
wangjksjtu@gmail.com

Jianing Zhou
Shanghai Jiao Tong University
Shanghai, China
zjn659644@sjtu.edu.cn

Gongshen Liu
Shanghai Jiao Tong University
Shanghai, China
lgshen@sjtu.edu.cn

Abstract

Chinese word segmentation (CWS) is often regarded as a character-based sequence labeling task in most current works which have achieved great performance by leveraging powerful neural networks. However, these works neglect an important clue: Chinese characters contain both semantic and phonetic meanings. In this paper, we introduce multiple character embeddings including *Pinyin Romanization* and *Wubi Input*, both of which are easily accessible and effective in depicting semantics of characters. To fully leverage them, we propose a novel shared Bi-LSTM-CRF model, which fuses multiple features efficiently. Extensive experiments on five corpora demonstrate that extra embeddings help obtain a significant improvement. Specifically, we achieve the state-of-the-art performance in AS and CityU datasets with F1 scores 96.9 and 97.3, respectively without leveraging any external resources.

Introduction

Different from English and other western languages of which the words are separated by white-space, Chinese is written without explicit word delimiters. In consequence, Chinese word segmentation (CWS) is a preliminary and essential pre-processing step for most other natural language processing (NLP) tasks, such as part-of-speech tagging (POS) and named-entity recognition (NER).

The representative approach is treating CWS as a character-based sequence labeling task following (Xue 2003) and (Peng, Feng, and McCallum 2004), trying to label every character with $\{B, M, E, S\}$ tagging scheme, which can be done by different neural networks including tensor neural network (Pei, Ge, and Chang 2014), gated recursive neural network (GRNN) (Chen et al. 2015a), long short-term memory neural network (LSTM) (Chen et al. 2015b) and convolutional neural network (Wang and Xu 2017).

It is worthy to notice that most of the neural network models depend heavily on the embeddings of Chinese characters. Since Mikolov et al. proposed word2vec techniques, the vector representation of words or characters has become an essential section in neural networks to solve NLP tasks in different languages.

However, existing approaches neglect an important clue: Chinese characters contain both semantic and phonetic information, that is, there are other effective representations of characters. The most intuitive one is *Pinyin Romanization* (拼音) that keeps a many-to-one relationship with Chinese characters. That is to say, for one character, different meanings in specific contexts may lead to different pronunciations. This phenomenon called *Polyphony* in linguistics is very common and crucial to word segmentation task. Apart from Pinyin Romanization, *Wubi Input* (五笔) is another effective representation which absorbs semantic meanings of Chinese characters. Compared to Radical (偏旁) (Sun et al. 2014; Dong et al. 2016; Shao et al. 2017), Wubi includes more comprehensive graphical and structural information that are highly relevant to the semantic meanings.

This paper will thoroughly study how important the extra embeddings are and what scholars can achieve by combining extra embeddings with representative models. To leverage extra phonetic and semantic information efficiently, we propose a shared Bi-LSTMs-CRF model, which feeds embeddings into three stacked LSTM layers with shared parameters and finally scores with CRF layer. Extensive experiments are conducted on five corpora. Our methods not only produce state-of-the-art results but are highly efficient in training.

In conclusion, the main contributions of this paper can be summarized as follows:

- We firstly propose to leverage both semantic and phonetic meanings of Chinese characters in NLP tasks by introducing Pinyin Romanization and Wubi Input embeddings, which are easily accessible and effective in representing semantic and phonetic features.
- We put forward multi-embedding architectures intended for CWS, which effectively fuse multiple embeddings, extract features and can be trained efficiently.
- Extensive experiments were conducted on Bakeoff2005 and CTB6 corpora. The results show that extra embeddings extract semantic and phonetic features effectively and help achieve the state-of-the-art performance.

Multiple Embeddings

Representing the symbolic data as distributed vectors is the very first step of using the neural networks to solve the CWS problem. The distributed vectors are also called embeddings

Preprint. Work in progress.

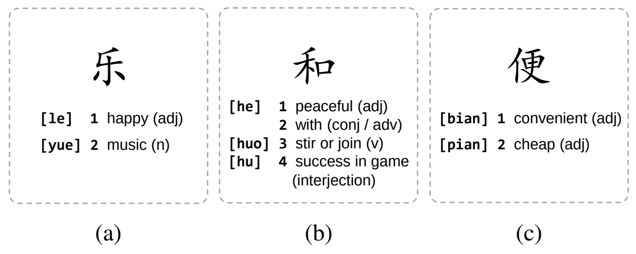


Figure 1: Examples of phono-semantic compound characters and polyphone characters.

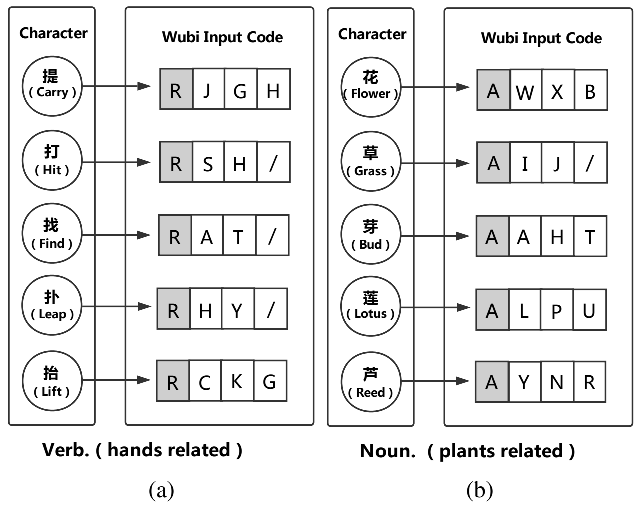


Figure 2: The potentially semantic relationships between Chinese characters and Wubi Input. Gray area indicates that these characters have the same first letter in the Wubi Input representation.

according to (Bengio, Vincent, and Janvin 2006) and (Collobert and Weston 2008).

To fully leverage the multiple properties of Chinese characters, we introduce three types of embeddings to extract three-dimension features: character embeddings for character-level features, Pinyin Romanization embeddings for phonetic features and Wubi Input embeddings for structure-level features.

Chinese Characters

Chinese word segmentation is often regarded as a character-based sequence labeling task, which aims to label every character with $\{B, M, E, S\}$ tagging scheme. Therefore, it is widely accepted that characters are the most fundamental inputs for neural networks (Chen et al. 2015b; Cai and Zhao 2016; Cai et al. 2017). In this paper, characters are still the basic inputs with two other presentations (*Pinyin* and *Wubi*) introduced in the following sections.

Pinyin Romanization

Pinyin Romanization (拼音) is the official romanization system for standard Chinese characters (ISO 7098:2015(E)

English Letters	Stoke	Examples
A, S, D, F, G	horizontal (一)	土, 木, 工
H, J, K, L, M	vertical (丨)	日, 田, 山
Q, W, E, R, T	left-falling (丿)	人, 月, 金
Y, U, I, O, P	right-falling (丶)	立, 火, 之
X, C, V, B, N	hook	已, 子, 女

Table 1: The map between encoding letters and their corresponding stroke types (ISO 7098:2015(E) 2015).

2015), which is established and normalized for distinguishing the characters of multiple meanings when people are communicating face to face. Pinyin represents the pronunciation of Chinese characters like phonogram in English. What is more, Pinyin often has close relationship with the semantics. That is to say, one character may have varied Pinyin codes which indicate different meanings. This phenomenon is very common in Asian languages and termed as polyphony.

Figure 1 shows some examples of polyphone characters. For instance, the character '乐' in Figure 1a has two different pronunciations (Pinyin codes). When pronounced as 'yue', it is used as a noun which means music. However, while it is pronounced as 'le', it indicates the happiness. Similarly, the character '便' in Figure 1c also has two meanings with varied Pinyin codes. When the code is 'bian', the character denotes something is convenient. On the contrary, the character represents something is cheap when the corresponding Pinyin code is 'pian'.

Through Pinyin codes, a natural bridge is constructed between the words and their semantics. Now that the humans could understand the different meanings of words by varied pronunciations, the neural networks are also likely to learn the mappings between semantic meanings and Pinyin codes automatically.

Wubi Input

Wubi Input (五笔) is also intended for depicting potentially semantic relationships between characters and focuses on the structure of characters. It is beneficial to CWS task mainly in two aspects: 1) Wubi codes capture the semantic meanings of characters; 2) characters with similar structures are more likely to make up a word.

Wubi is an efficient encoding system which represents each Chinese character with at most four English letters. Specifically, these letters are divided into five regions, each of which represents a type of structure (stroke, 笔画) in Chinese characters. The encoding map is given in Table 1.

Figure 2 gives some examples and shows the potential semantic relationships between Chinese characters and Wubi Inputs. Each character is encoded to the solely corresponding Wubi code (four letters), which indicates the structures and the meanings of the characters. For instance, '提' (carry), '打' (hit) and '抬' (lift) in Figure 2a are all verbs which are related to hands but have completely different spellings in English. However, in Chinese, their corresponding characters are all left-right symbols and have the same radical (‘R’ in Wubi codes). That is to say, characters that are highly relevant in semantics usually have similar structures which could be perfectly captured by Wubi. Besides, characters with similar structures are more likely to make up a word. For example, ‘花’ (flower), ‘草’ (grass) and ‘芽’ (bud) in Figure 2b are nouns that indicate plants but have irrelevant spellings in English. Whereas, they are all up-down symbols and have the same radical (‘A’ in Wubi codes). More importantly, these words usually make up new words such as ‘花草’ (flowers and grasses) and ‘花芽’ (the buds of flowers).

In addition, the sequence in Wubi codes is one approach of interpreting the relationships between Chinese characters. As shown in Figure 3, it is easy to find some interesting rules. For instance, we could conclude: 1) the sequence order implies the orders of components (e.g., ‘IA’ vs ‘AI’ and ‘IY’ vs ‘YI’); 2) some code has practical meanings (e.g., ‘I’ means water). In consequence, Wubi is an efficient description of the Chinese characters.

Multiple Embeddings

To fully utilize the various properties of Chinese characters, we construct the Pinyin and Wubi Embeddings before feeding character embeddings into neural networks directly. We firstly preprocess the characters following the strategy in (Lample et al. 2016; Shao et al. 2017). Second, we use the Pypinyin Library[1] to annotate Pinyin codes. Third, an official transformation table[2] is leveraged to translate characters into Wubi codes. Finally, we retrieve multiple embeddings using word2vec tool (Mikolov et al. 2013).

General Neural Networks for CWS

We adopt the popular Bi-LSTMs-CRF as our baseline model (Figure 3), similar to the architectures proposed by (Lample et al. 2016) and (Dong et al. 2016).

LSTM Unit

The fundamental unit in Bi-LSTM network is LSTM memory unit composed of three essential gates and one cell state.

As shown in Equation (1), the core of LSTM unit is the memory cell $\mathbf{c}$ containing the encoded information memorized at every time step t. The state of the memory cell is bound up with three essential gates, namely input gate $\mathbf{i}$, forget gate $\mathbf{f}$ and output gate $\mathbf{o}$. Each gate is composed of the input part and recurrent part.

If the gate is non-zero vectors, the gate will be capable of scaling input values. If the gate is zero, it will omit the inputs. The output gate will feed its output that is previous hidden state and input of the upper layer of the neural network at the current time step t into the next time step $t + 1$.

The definitions and formulas of each gate, cell state and

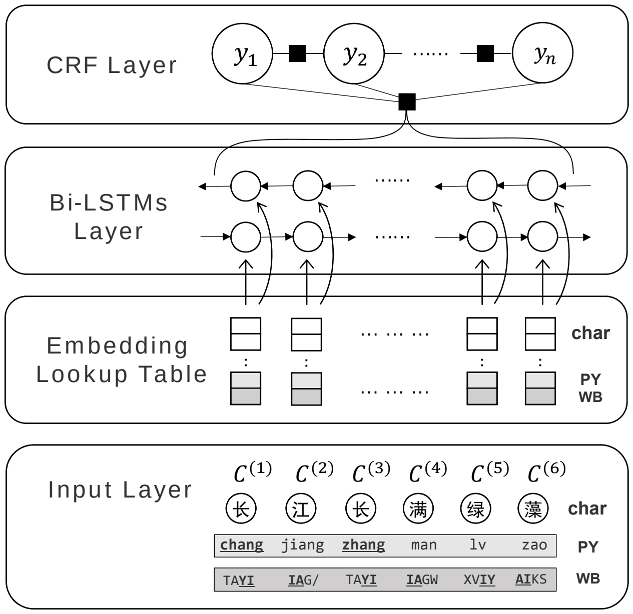

CRF Layer
y_1
y_2
......
y_n
Bi-LSTMs Layer
......
Embedding Lookup Table
char
PY
WB
Input Layer
$C^{(1)}$ $C^{(2)}$ $C^{(3)}$ $C^{(4)}$ $C^{(5)}$ $C^{(6)}$
长 江 长 满 绿 藻
char
chang jiang zhang man lv zao
PY
TAYI IAG/ TAYI IAGW XVIY AIKS
WB

Figure 3: The architecture of Bi-LSTM-CRF network. PY and WB represent *Pinyin Romanization* and *Wubi Input* introduced in this paper.

output are defined as follows:

$$\begin{aligned}
\mathbf{i}^{(t)} &= \sigma(\mathbf{W}_{ix}\mathbf{x}^{(t)} + \mathbf{W}_{ih}\mathbf{h}^{(t-1)} + \mathbf{W}_{ic}\mathbf{c}^{(t-1)}),\\
\mathbf{f}^{(t)} &= \sigma(\mathbf{W}_{fx}\mathbf{x}^{(t)} + \mathbf{W}_{fh}\mathbf{h}^{(t-1)} + \mathbf{W}_{fc}\mathbf{c}^{(t-1)}),\\
\mathbf{c}^{(t)} &= \mathbf{i}^{(t)} \odot \phi(\mathbf{W}_{cx}\mathbf{x}^{(t)} + \mathbf{W}_{ch}\mathbf{h}^{(t-1)})\\
&\quad + \mathbf{f}^{(t)} \odot \mathbf{c}^{(t-1)},\\
\mathbf{o}^{(t)} &= \sigma(\mathbf{W}_{ox}\mathbf{x}^{(t)} + \mathbf{W}_{oh}\mathbf{h}^{(t-1)} + \mathbf{W}_{oc}\mathbf{c}^{(t)}),\\
\mathbf{h}^{(t)} &= \mathbf{o}^{(t)} \odot \phi(\mathbf{c}^{(t)}).
\end{aligned} \tag{1}$$

Here σ is the logistic sigmoid function and ϕ is the hyperbolic tangent function; $\mathbf{i}^{(t)}$, $\mathbf{f}^{(t)}$, $\mathbf{o}^{(t)}$ and $\mathbf{c}^{(t)}$ are respectively the input gate, forget gate, output gate and memory cell activation vector at time step t; $\mathbf{x}^{(t)} \in \mathbb{R}^{H_2}$ denotes the input vector at time step t; All vectors have the same size as the hidden vector $\mathbf{h}^{(t)} \in \mathbb{R}^{H_2}$; $\mathbf{W}$ denotes all square matrices with different subscripts; $\odot$ denotes the element-wise multiplication of the vectors.

Bi-LSTM Network

The Bi-LSTM has two parallel layers in both propagation directions. Each character in a given sentence $(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)$ is represented as a $\mathbf{H}_2$-dimensional vector.

An LSTM generates a representation $\overrightarrow{\mathbf{h}^{(t)}}$ of the left context of the sentence at every character $\mathbf{t}$. Moreover, computing a representation $\overleftarrow{\mathbf{h}^{(t)}}$ of the right context of the sentence at every character *t* should also add some useful information, which can be achieved using a second LSTM that reads the same sentence in reverse. The former is referred to as the forward LSTM and the latter is referred to as the backward

[1]https://pypi.python.org/pypi/pypinyin

[2]http://wubi.free.fr/index_en.html

LSTM. This forward LSTM and backward LSTM pair is referred to as a Bi-LSTM.

The final representation of a word in a sentence using Bi-LSTM model is generated by concatenating the left and the right context representation, that is, $\mathbf{h}^{(t)} = [\overrightarrow{\mathbf{h}^{(t)}};\overleftarrow{\mathbf{h}^{(t)}}]$.

Stacked Bi-LSTM

The Stacked Bi-LSTM is an extension of Bi-LSTM, which is created by stacking multiple LSTM hidden layers on top of each other. The input sequence of one layer is formed by the output sequence of the upper layer. For instance, three LSTM layers cases are demonstrated in Equation (2):

$$\begin{aligned} \mathbf{h}_1^{(t)} &= \overrightarrow{\mathbf{h}_1^{(t)}} + \overleftarrow{\mathbf{h}_1^{(t)}}, \\ \mathbf{h}_2^{(t)} &= \overrightarrow{\mathbf{h}_2^{(t)}} + \overleftarrow{\mathbf{h}_2^{(t)}}, \\ \mathbf{h}_3^{(t)} &= [\overrightarrow{\mathbf{h}_3^{(t)}}; \overleftarrow{\mathbf{h}_3^{(t)}}], \end{aligned} \tag{2}$$

where $\mathbf{h}_1^{(t)}$ is the input sequence of the second LSTM layer, $\mathbf{h}_2^{(t)}$ is the input sequence of the third LSTM layer and $\mathbf{h}_3^{(t)}$ is the final output of the Bi-LSTM$\times 3$ network.

CRF layer

CWS is usually regarded as a sequence-labeling task. Each character in a sentence is labeled as one of $\{B, M, E, S\}$, which indicate the beginning of a word, middle of a word, end of a word and a word with single character, respectively.

Generally, given a sequence $X = \{x_1, \ldots, x_n\}$, the aim of the character sequence labeling task is to figure out labels Y^* = $\{y_1^*, \ldots, y_n^*\}$:

$$Y^* = \underset{Y \in \mathcal{L}^n}{\arg\max}\, p(Y|X), \tag{3}$$

where $\mathcal{L} = \{B, M, E, S\}$.

Specifically, conditional random fields (CRF) layer is employed here to score the labels. In CRF layer, $p(Y|X)$ could be formalized as follows:

$$p(Y|X) = \frac{\Psi(Y|X)}{\sum_{Y' \in \mathcal{L}^n} \Psi(Y'|X)} \tag{4}$$

where $\Psi(Y|X)$ is the potential function. Moreover, we only put interactions between two successive labels into consideration:

$$\begin{aligned} \Psi(Y|X) &= \prod_{i=2}^{n} \psi(X, i, y_{i-1}, y_i), \\ \psi(\mathbf{x}, i, y', y) &= exp(s(X, i)_y + \mathbf{b}_{y'y}), \end{aligned} \tag{5}$$

where $\mathbf{b}_{y'y} \in \mathbf{R}$ are trainable parameters with respective to label pair (y', y); $s(X, i) \in \mathbb{R}^{|\mathcal{L}|}$ is a score function assigning score for each label on tagging the i-th character:

$$s(X, i) = \mathbf{W_s}^\top \mathbf{h_i} + \mathbf{b_s}, \tag{6}$$

where $\mathbf{h_i}$ is the hidden state of Bi-LSTM at position i; $\mathbf{W_s} \in \mathbb{R}^{d_h \times |\mathcal{L}|}$ and $\mathbf{b_s} \in \mathbb{R}^{|\mathcal{L}|}$ are trainable parameters.

Multi-Embedding Model Architecture

To utilize multiple embeddings and represent the semantic and phonetic meanings effectively and efficiently, we have designed three novel architectures shown in Figure 4.

Model-I adopts a fully connected layer to merge three kinds of embeddings together. Model-II feeds multiple embeddings into three independent stacked Bi-LSTMs networks. Model-III, however, feeds three kinds of embeddings into a shared Bi-LSTMs network one by one.

Model-I: FC-Layer Bi-LSTMs-CRF Model

As is shown in Figure 4a, we adopt a fully connected layer to merge three input vectors in Model-I: character embeddings, Pinyin embeddings and Wubi embeddings:

$$\begin{aligned} \mathbf{x}_{in}^{(t)} &= [\mathbf{x}_c^{(t)}; \mathbf{x}_p^{(t)}; \mathbf{x}_w^{(t)}], \\ \mathbf{x}^{(t)} &= \sigma(\mathbf{W}_{fc}\mathbf{x}_{in}^{(t)} + \mathbf{b}_{fc}), \end{aligned} \tag{7}$$

where σ is the logistic sigmoid function; $\mathbf{W}_{fc}$ and $\mathbf{b}_{fc}$ are trainable parameters of fully connected layer; $\mathbf{x}_c^{(t)}$, $\mathbf{x}_p^{(t)}$ and $\mathbf{x}_w^{(t)}$ are the input vectors of character, pinyin and wubi embeddings. The output of the fully connected layer $\mathbf{x}^{(t)}$ forms the input sequence of the Bi-LSTMs-CRF.

Model-II: Multi-Bi-LSTMs-CRF Model

In Model-II shown in Figure 4b, the input vectors of character, pinyin and wubi embeddings are fed into stacked Bi-LSTMs networks respectively:

$$\begin{aligned} \mathbf{h}_{3,c}^{(t)} &= \text{Bi-LSTMs}_1(\mathbf{x}_c^{(t)}, \theta_c), \\ \mathbf{h}_{3,p}^{(t)} &= \text{Bi-LSTMs}_2(\mathbf{x}_p^{(t)}, \theta_p), \\ \mathbf{h}_{3,w}^{(t)} &= \text{Bi-LSTMs}_3(\mathbf{x}_w^{(t)}, \theta_w), \\ \mathbf{h}^{(t)} &= \mathbf{h}_{3,c}^{(t)} + \mathbf{h}_{3,p}^{(t)} + \mathbf{h}_{3,w}^{(t)}, \end{aligned} \tag{8}$$

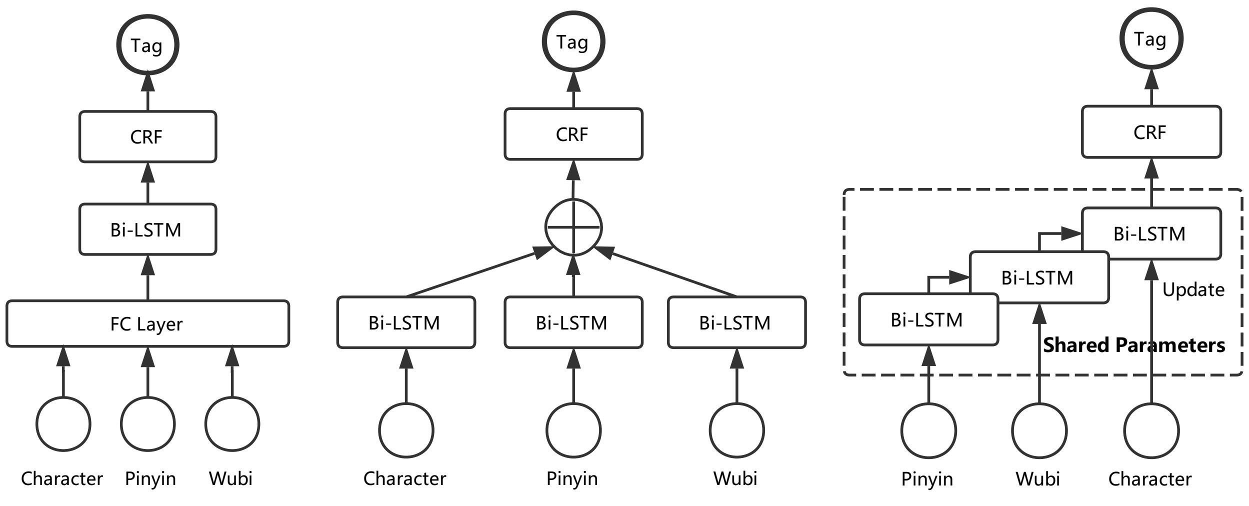
where θ_c, θ_p and θ_w denote parameters in three Bi-LSTMs networks respectively. The outputs of three-layer Bi-LSTMs are $\mathbf{h}_{3,c}^{(t)}$, $\mathbf{h}_{3,p}^{(t)}$ and $\mathbf{h}_{3,w}^{(t)}$, which form the inputs of the CRF layer $\mathbf{h}_{(t)}$.

Model-III: Shared Bi-LSTMs-CRF Model

As for Model-III, we feed character, Pinyin and Wubi embeddings into a stacked Bi-LSTMs network shared with the same parameters, which is shown in Figure 4c:

$$\begin{aligned} \begin{bmatrix} \mathbf{h}_{3,c}^{(t)} \\ \mathbf{h}_{3,p}^{(t)} \\ \mathbf{h}_{3,w}^{(t)} \end{bmatrix} &= \text{Bi-LSTMs}(\begin{bmatrix} \mathbf{w}_c^{(t)} \\ \mathbf{w}_p^{(t)} \\ \mathbf{w}_w^{(t)} \end{bmatrix}, \theta), \\ \mathbf{h}^t &= \mathbf{h}_{3,c}^{(t)} + \mathbf{h}_{3,p}^{(t)} + \mathbf{h}_{3,w}^{(t)}, \end{aligned} \tag{9}$$

where θ denotes the shared parameters of Bi-LSTMs. Different from Equation (8), there is only one shared Bi-LSTMs rather than three independent Bi-LSTMs, which process much more trainable parameters. As a consequence, the shared Bi-LSTMs-CRF model is trained more efficiently.



(a) Model-I (b) Model-II (c) Model-III

Figure 4: Network architecture of three multi-embedding models.

Experimental Evaluations

In this section, we provide empirical results to confirm that multiple embeddings indeed lead to a significant improvement for CWS. Besides, our novel model (Model-III) could be trained very efficiently (almost no cost than baseline) as well as obtaining the state-of-the-art performance.

Experimental Setup

Corpora To make the results comparable and convincing, we evaluate our models on SIGHAN 2005 (Emerson 2005) and Chinese Treebank 6.0 (CTB6) (Xue et al. 2005) datasets, which are widely used in previous works. We split the data into training, development, test set following the official documents and use standard word precision (P), recall (R) and F1 measure (F) (Emerson 2005). The detailed statistics of five corpora are summarized in Table 2.

Preprocessing We leverage standard word2vec tools provided by (Mikolov et al. 2013) to obtain multiple embeddings and established look-up tables. Besides, the traditional Chinese corpora (AS and CityU) are transformed to the simplified corpora before training like (Chen et al. 2017). In our experiments, the unknown characters are mapped to a special symbol $<UNK>$, and the maximal length of sentences is set to 80 (Lample et al. 2016).

Training Hyper-parameters of models, such as embedding size, learning rate, batch size and the number of Bi-LSTM layers, make sense in the performance of the models. To get competitive results, we fine-tune the parameters properly. In the experiments of this paper, we explore two determinant factors, namely, the embedding size and the number of LSTM layers like (Yao and Huang 2016).

N-gram Features Previous works (Pei, Ge, and Chang 2014; Chen et al. 2015a; 2015b; Sun, Shen, and Deng 2017) show that *n*-gram features are beneficial in CWS. To build a stronger baseline and also achieve better performance, we implement *n*-gram features function in our models, which are extracted by word2vec tool.

Dataset		#chars	#words	#Sents	OOV
CTB6	Train	1.1M	0.6M	23.4K	-
	Test	0.1M	0.1M	2.1K	5.55%
PKU	Train	1.8M	1.1M	47.3K	-
	Test	0.3M	0.2M	6.4K	3.61%
MSR	Train	4.1M	2.4M	86.9K	-
	Test	0.2M	0.1M	4.0K	2.60%
AS	Train	8.4M	5.4M	709.0K	-
	Test	0.2M	0.1M	14.4K	4.30%
CityU	Train	1.8M	1.1M	36.2K	-
	Test	0.3M	0.2M	6.7K	8.23%

Table 2: Statistics of training, development and test sets on five datasets.

Embedding Size	P	R	F
64	94.2	93.4	93.8
128	94.8	95.0	94.8
256	96.3	95.7	96.0
300	96.2	95.9	96.0

Table 3: Comparison results of Model-I employing different embedding sizes on PKU.

Experimental Results

Dominant Hyper-parameters To achieve competitive performance, it is necessary to fine-tune hyper-parameters of neural networks appropriately. Extensive experiments reveal that there are two dominant parameters, which affect the performance of our models greatly, namely, the embedding

Layers	baseline			Model-I		
	P	R	F	P	R	F
Bi-LSTM	92.7	91.6	92.1	93.8	93.5	93.6
Bi-LSTM×3	95.8	95.9	95.8	96.3	95.7	96.0
Bi-LSTM×4	95.2	95.1	95.2	95.9	95.8	95.9

Table 4: Comparison results of Model-I employing different number of LSTM layers on PKU.

Hyper-parameters	Setting
Multiple Embedding Size	**256**
Dropout rate	0.5
Initial learning rate	$5e^{-3}$
Hidden unit number	200
Bi-LSTM layers	**3**

Table 5: Hyper-parameters configuration of state-of-the-art models in the following experiments. Bold font represents most dominant factors.

size and the number of Bi-LSTM layers.

Table 3 and 4 demonstrate the influence of these two crucial parameters. From experiments, we find that an appropriate embedding size and number of Bi-LSTM layers help models achieve better performance. As a result, in the following experiments, we adopt the configuration given in Table 5.

Performance under Different Architectures We comprehensively conducted the analysis of three proposed architectures. As illustrated in Table 6, considerable improvements are obtained by three multi-embedding models compared with baseline model. Among three architectures, overall, Model-III (shared Bi-LSTMs-CRF) achieves better performance even if it processes fewer trainable parameters.

Embedding Ablation To explore the effectiveness of Pinyin and Wubi embeddings individually, embedding ablation experiments are conducted on different corpora. That is, we train models with Pinyin or Wubi embeddings separately. As shown in Table 7, the Pinyin and Wubi embeddings have brought significant improvements compared to baseline model separately.

Competitive Performance To validate the effectiveness of extra embeddings for CWS, we compare our models with previous state-of-the-art models.

Table 8 shows the comprehensive comparison of performance on all Bakeoff2005 corpora. To the best of our knowledge, we have achieved the best performance on AS and CityU dataset (with F1 score 96.9 and 97.3 respectively) and competitive performance on PKU and MSR even if not leveraging external resources (such as pre-trained char/word embeddings, extra dictionaries, labelled or unlabelled corpora). In consequence, multiple character embeddings benefit a lot for CWS.

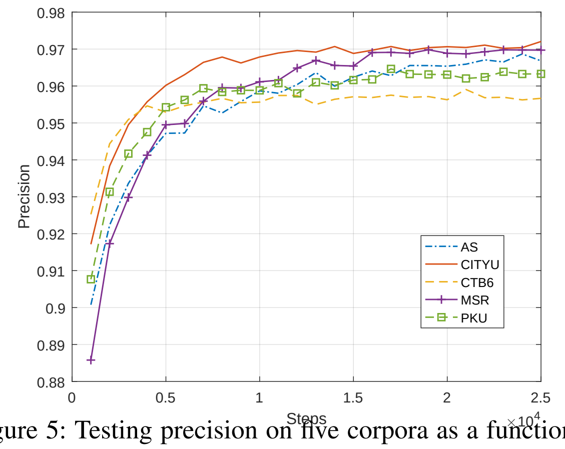


Figure 5: Testing precision on five corpora as a function of training steps.

Convergence Speed In this paper, we curve the labelling precision trends with training steps increasing, in which one step denotes one batch size training. As shown in Figure 5, the convergence speed is similar in five corpora. After around 1×10^4 steps (usually smaller than 100 epoches, 4 hours in GTX 1080), the model reaches convergence.

To further explore the speed performance, we gather the statistics of proposed models' training time per batch and convergent time in Table 9. Compared to the baseline model, Model-III almost consume the same training time $(1.07\times)$ per batch and convergence time $(1.04\times)$ but obtains a considerable improvement. In consequence, we recommend Model-III in practice for high efficiency.

Related Work

Since (Xue 2003), researchers have mostly treated CWS as a sequence labeling problem. Following this idea, great achievements have been reached in the past few years with the effective embeddings introduced and powerful neural networks armed.

In recent years, there are plentiful works exploiting different neural network architectures in CWS. Among these architectures, there are several models most similar to our model: Bi-LSTM-CRF (Huang, Xu, and Yu 2015), Bi-LSTM-CRF (Lample et al. 2016; Dong et al. 2016), and Bi-LSTM-CNNs-CRF (Ma and Hovy 2016).

Huang, Xu, and Yu was the first to adopt Bi-LSTM neural network for character representations and CRF for label decoding. Lample et al. and Dong et al. both exploited the Bi-LSTM-CRF for named entity recognition. Separately, Lample et al. mainly focused on English, German, Dutch and Spanish and Dong et al. focused on Chinese. Moreover, Dong et al. utilized radical-level information, which is similar to the character-structure-level information utilized in our model.

(Ma and Hovy 2016) introduced a novel neural network architecture by combining Bi-LSTM, CNN and CRF, which results in faster convergence speed and better performance on POS and NER tasks. In addition, their model leverages both the character-level and word-level information.

Models	CTB6			PKU			MSR			AS			CityU		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
baseline	94.1	94.0	94.1	95.8	95.9	95.8	95.3	95.7	95.5	95.6	95.5	95.6	95.9	96.0	96.0
Model-I	**95.4**	**95.3**	**95.4**	**96.3**	95.7	96.0	96.6	96.5	96.6	96.8	96.5	96.7	**97.2**	**97.0**	**97.1**
Model-II	94.9	95.0	94.9	95.7	95.7	95.7	96.8	96.6	96.7	96.6	96.5	96.5	96.7	96.5	96.6
Model-III	**95.4**	95.0	95.2	**96.3**	**96.1**	**96.2**	**97.0**	**96.9**	**97.0**	**96.9**	**96.8**	**96.9**	97.1	**97.0**	**97.1**

Table 6: Comparison of different architectures on all of five corpora. In most corpora, Model-III achieves better performance. All of three proposed models have a significant improvement compared with baseline model.

Models	CTB6			PKU		
	P	R	F	P	R	F
baseline	94.1	94.0	94.1	95.8	95.9	95.8
IO + PY	94.6	94.9	94.8	96.0	95.6	95.9
IO + WB	95.3	**95.4**	95.3	96.2	**96.1**	**96.2**
Model-I	**95.4**	95.3	**95.4**	**96.3**	95.7	96.0

Table 7: Embedding ablation results on CTB and PKU. IO + PY and IO + WB denote injecting Pinyin and Wubi embeddings separately under Model-I architecture.

Model	PKU	MSR	AS	CityU
(Tseng et al. 2005)	95.0	96.4	-	-
(Zhang and Clark 2007)	95.0	96.4	-	-
(Zhao and Kit 2008)	95.4	97.6	96.1	95.7
(Sun et al. 2009)	95.2	97.3	-	-
(Sun and Wan 2012)	95.4	97.4	-	-
(Chen et al. 2015b)	94.8	95.6	-	-
(Chen et al. 2017)	94.3	96.0	-	94.8
(Zhang et al. 2013)*	96.1	97.4	-	-
(Chen et al. 2015b)*	96.5	97.4	-	-
(Cai et al. 2017)*	95.8	97.1	95.6	95.3
(Wang and Xu 2017)*	**96.5**	**98.0**	-	-
(Sun, Shen, and Deng 2017)*	96.0	97.9	96.1	96.9
baseline	95.8	95.5	95.6	96.0
ours (+PY)	96.0	96.8	96.7	97.0
ours (+WB)	**96.3**	97.2	96.5	**97.3**
ours (+PY+WB)	96.2	97.0	**96.9**	97.1

Table 8: Comparison with previous state-of-the-art models on all four Bakeoff2005 datasets (**F1 score**). Results with $*$ represent allowing the use of external resources. Note that our approach doesn't leverage any external resources.

Our work distinguishes itself by utilizing multiple dimensions of features in Chinese characters. With phonetic and semantic meanings taken into consideration, three proposed models achieve better performance on CWS and could be applied in POS and NER tasks. In particular, compared to radical-level information in (Dong et al. 2016), Wubi Input processes richer structure details and potentially semantic relationships.

Recently, researchers propose to regard CWS as a word-based sequence labeling problem, which also achieve competitive performance (Zhang, Zhang, and Fu 2016; Cai and Zhao 2016; Cai et al. 2017; Yang, Zhang, and Dong 2017). Other works try to introduce very deep networks (Wang and Xu 2017) or treat CWS as a gap-filling problem (Sun, Shen, and Deng 2017). We believe that our multiple embeddings could also be transferred into word-level and improve the performance. In a nutshell, multiple embeddings are generic and easily accessible, which can be applied and studied further in these works.

Model	Time (batch)	Time (P-95%)
baseline	1 ×	1 ×
Model-I	1.03 ×	1.50 ×
Model-II	2.61 ×	2.51 ×
Model-III	**1.07** ×	**1.04** ×

Table 9: Relative training time of document models on MSR. The left column is average batch-size training time and the right column is convergence time, where over 95% accuracy in test is considered as convergence.

Conclusion and Future Work

In this paper, we firstly propose to leverage both semantic and phonetic meanings of Chinese characters by introducing multiple character embeddings in NLP tasks. We conduct a comprehensive analysis on why Pinyin and Wubi embeddings are so essential in CWS task and could be translated to other NLP tasks. Besides, to fully leverage multiple embeddings, we design three generic models to fuse extra features and produce the start-of-the-art performance in five public corpora. In particular, the shared Bi-LSTMs-CRF models could be trained efficiently and meanwhile, produce the best performance on AS and CityU with 0.8% and 0.4% F score increase, respectively.

Although our paper shows that multiple embeddings result in significant improvements in performance, we believe that there are other effective ways to leverage them due to complex hierarchies in Chinese. Besides, multiple embeddings may be helpful in other NLP tasks and could be translated into other Asian languages. Finally, The extra embeddings may also help the refinement process, which means, utilizing Pinyin or Wubi to revise the mis-labelled sentences. We leave these points to further works.

References

Bengio, Y.; Vincent, P.; and Janvin, C. 2006. A neural probabilistic language model. *Journal of Machine Learning Research* 3(6):1137–1155.

Cai, D., and Zhao, H. 2016. Neural word segmentation learning for chinese. In *ACL (1)*. The Association for Computer Linguistics.

Cai, D.; Zhao, H.; Zhang, Z.; Xin, Y.; Wu, Y.; and Huang, F. 2017. Fast and accurate neural word segmentation for chinese. In *ACL (2)*, 608–615. Association for Computational Linguistics.

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

Chen, X.; Qiu, X.; Zhu, C.; Liu, P.; and Huang, X. 2015b. Long short-term memory neural networks for chinese word segmentation. In *EMNLP*, 1197–1206. The Association for Computational Linguistics.

Chen, X.; Shi, Z.; Qiu, X.; and Huang, X. 2017. Adversarial multi-criteria learning for chinese word segmentation. In *ACL (1)*, 1193–1203. Association for Computational Linguistics.

Collobert, R., and Weston, J. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *International Conference on Machine Learning*, 160–167.

Dong, C.; Zhang, J.; Zong, C.; Hattori, M.; and Di, H. 2016. Character-based LSTM-CRF with radical-level features for chinese named entity recognition. In *NLPCC/ICCPOL*, volume 10102 of *Lecture Notes in Computer Science*, 239–250. Springer.

Emerson, T. 2005. The second international chinese word segmentation bakeoff. In *SIGHAN@IJCNLP 2005*. ACL.

Huang, Z.; Xu, W.; and Yu, K. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR* abs/1508.01991.

2015. Information and documentation – Romanization of Chinese. Standard, International Organization for Standardization, Geneva, CH.

Lample, G.; Ballesteros, M.; Subramanian, S.; Kawakami, K.; and Dyer, C. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Ma, X., and Hovy, E. H. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *CoRR* abs/1603.01354.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, 3111–3119.

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

Peng, F.; Feng, F.; and McCallum, A. 2004. Chinese segmentation and new word detection using conditional random fields. In *COLING*.

Shao, Y.; Hardmeier, C.; Tiedemann, J.; and Nivre, J. 2017. Character-based joint segmentation and POS tagging for chinese using bidirectional RNN-CRF. In *IJCNLP(1)*, 173–183. Asian Federation of Natural Language Processing.

Sun, W., and Wan, X. 2012. Reducing approximation and estimation errors for chinese lexical processing with heterogeneous annotations. In *ACL (1)*, 232–241. The Association for Computer Linguistics.

Sun, X.; Zhang, Y.; Matsuzaki, T.; Tsuruoka, Y.; and Tsujii, J. 2009. A discriminative latent variable chinese segmenter with hybrid word/character information. In *HLT-NAACL*, 56–64. The Association for Computational Linguistics.

Sun, Y.; Lin, L.; Yang, N.; Ji, Z.; and Wang, X. 2014. Radical-enhanced chinese character embedding. In *ICONIP (2)*, volume 8835 of *Lecture Notes in Computer Science*, 279–286. Springer.

Sun, Z.; Shen, G.; and Deng, Z. 2017. A gap-based framework for chinese word segmentation via very deep convolutional networks. *CoRR* abs/1712.09509.

Tseng, H.; Chang, P.; Andrew, G.; Jurafsky, D.; and Manning, C. D. 2005. A conditional random field word segmenter for sighan bakeoff 2005. In *SIGHAN@IJCNLP 2005*. ACL.

Wang, C., and Xu, B. 2017. Convolutional neural network with word embeddings for chinese word segmentation. *CoRR* abs/1711.04411.

Xue, N.; Xia, F.; Chiou, F.; 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 (1)*, 839–849. Association for Computational Linguistics.

Yao, Y., and Huang, Z. 2016. Bi-directional LSTM recurrent neural network for chinese word segmentation. In *ICONIP (4)*, volume 9950 of *Lecture Notes in Computer Science*, 345–353.

Zhang, Y., and Clark, S. 2007. Chinese segmentation with a word-based perceptron algorithm. In *ACL*. The Association for Computational Linguistics.

Zhang, L.; Wang, H.; Sun, X.; and Mansur, M. 2013. Exploring representations from unlabeled data with co-training for chinese word segmentation. In *EMNLP*, 311–321. ACL.

Zhang, M.; Zhang, Y.; and Fu, G. 2016. Transition-based neural word segmentation. In *ACL (1)*. The Association for Computer Linguistics.

Zhao, H., and Kit, C. 2008. Unsupervised segmentation helps supervised learning of character tagging for word segmentation and named entity recognition. In *IJCNLP*, 106–111. The Association for Computer Linguistics.