A Feature-Enriched Neural Model for Joint Chinese Word Segmentation and Part-of-Speech Tagging

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

Abstract

Recently, neural network models for natural language processing tasks have been increasingly focused on for their ability of alleviating the burden of manual feature engineering. However, the previous neural models cannot extract the complicated feature compositions as the traditional methods with discrete features. In this work, we propose a feature-enriched neural model for joint Chinese word segmentation and part-of-speech tagging task. Specifically, to simulate the feature templates of traditional discrete feature based models, we use different filters to model the complex compositional features with convolutional and pooling layer, and then utilize long distance dependency information with recurrent layer. Experimental results on five different datasets show the effectiveness of our proposed model.

1 Introduction

Chinese word segmentation and part-of-speech (POS) tagging are two core and fundamental tasks in Chinese natural language processing (NLP). The state-of-the-art approaches are based on joint segmentation and tagging (S&T) model, which can be regarded as character based sequence labeling task. The joint model can alleviate the error propagation problem of pipeline models.

Previously, the traditional hand-crafted feature based models have achieved great success on joint S&T task [Jiang et al., 2008; Kruengkrai et al., 2009; Qian et al., 2010; Zhang and Clark, 2008; 2010]. Despite of their success, their performances are easily affected by following two limitations.

The first is model complexity. Since the decoding space of joint S&T task is relatively large, the traditional models often rely on millions of discrete features. Therefore, the efficiency of joint S&T models is rather low. Moreover, these models suffer from data sparsity. Recently, some neural models [Huang et al., 2015; Chen et al., 2015a; Ma and Hovy, 2016] are proposed to reduce the efforts of feature engineering and the model complexity. However, these neural models just concatenate the embeddings of the context characters, and feed them into neural network. Since the concatenation operation is relatively simple, it is difficult to model the complicated features as the traditional discrete feature based models. Although the complicated interactions of inputs can be modeled by the deep neural network, the previous neural models show that the deep model cannot outperform the one with a single non-linear model.

The second is long term dependency. Unlike pure POS tagging task which can utilize contextual features on word level, joint S&T task usually extracts the contextual features on character level. Thus, the joint model need longer dependency on character level. As the example shown in Figure 1, conditional random field (CRF) model makes mistakes on words “改革 (reform)” and “精简 (simplify)” since it is hard for CRF to disambiguate the POS tags without using long distance information. However, restricted by model complexity and data sparsity, a larger window size (greater than 5) will instead hurt the performance. Therefore, how to exploit the long distance information without increasing the model complexity is crucial to joint S&T task.

In order to address these two problems, we propose a feature-enriched neural model for joint S&T task, which consists of several key components: (1) a convolutional layer to simulate compositional features as complex hand-crafting features; (2) a pooling layer to select the most valuable features; (3) a bi-directional long short-term memory (BLSTM) layer on the top to carry long distance dependency information. In addition, we introduce a highway layer [Srivastava et al., 2015] to increase the depth of architecture and obtain more sophisticated feature representation without suffering from the problem of gradient vanishing, leading to fast convergence.

Our contributions could be summaries as follows:
1. We propose a customized neural architecture for joint

* Corresponding author.
S&T task, in which each component is designed according to its specific requirements, instead of a general deep neural model.

2. Our model can alleviate two crucial problems: model complexity and long term dependency in joint S&T task.

3. We evaluate our model on five different datasets. Experimental results show that our model achieves comparable performance to the previous sophisticated feature based models, and outperforms the previous neural models.

2 Neural Models for Joint S&T

The joint S&T task is usually regarded as a character based sequence labeling problem.

In this paper, we employ the \{B M E S\} tag set $T_{SEG}$ (indicating the Begin, Middle, End of the word, and a Single character word respectively) for word segmentation and the tag set $T_{POS}$ (varies from dataset to dataset) for POS tagging.

The tag set $T$ of our joint S&T task would be the cross-label set of $T_{SEG}$ and $T_{POS}$. As illustrated in Figure 1, we would have a tag B_VV for character “结”，where B $\in T_{SEG}$ and VV $\in T_{POS}$, indicating the first character of the VV word “结合”.

Conventional neural network based model for sequence labeling task usually consists of three phases. Figure 2 gives the illustration.

2.1 Lookup Table Phase

In order to represent characters as distributed vectors, we usually apply a feed-forward neural layer on the top of the one-hot character representations. The parameter matrix of the neural layer is called character embedding matrix $E \in \mathbb{R}^{C \times d}$, where $C$ is the character set and $d$ is the dimensionality of the character embeddings. For a given sentence $c_{1:n}$ of length $n$, the first step is to lookup embeddings of the characters in the current window slide $c_{i-2} \cdots c_i : c_{i+1} \cdots c_{i+2}$ for the current character $c_i$ which is going to be tagged, where $k$ is a hyper-parameter indicating the window size. By concatenating the embeddings, we get the representation $x_i$ for the current character $c_i$.

2.2 Encoding Phase

Usually, we apply a linear transformation followed by a non-linear function to the current input $x_i$:

$$h_i = g(W_h^T \times x_i + b_h),$$

where $W_h \in \mathbb{R}^{kd \times h}$ and $b_h \in \mathbb{R}^h$ is the trainable parameters, and $h$ is the dimensionality of the hidden layer, $g(\cdot)$ is a non-linear function which could be $\text{sigmoid}(\cdot)$, $\text{tanh}(\cdot)$, etc.

Then, we could get the score vector $p_i \in \mathbb{R}^{T|} for each possible tags of current character $c_i$ by applying a linear transformation layer to the hidden layer $h_i$:

$$p_i = W_p^T \times h_i + b_p,$$

where $W_p \in \mathbb{R}^{h \times |T|}$ and $b_p \in \mathbb{R}^{|T|}$ is the trainable parameters, and $T$ is the joint tag set.

2.3 Decoding Phase

The decoding phase aims to select the best tag sequence $\hat{t}_{1:n}$, to maximize the reward function $r(\cdot)$:

$$r(t_{1:n}) = \sum_{i=2}^{n} (A_{t_{i-1}, t_i}) + \sum_{i=1}^{n} (p_i[t_i]),$$

$$\hat{t}_{1:n} = \text{arg max} \ r(t_{1:n}),$$

where $A \in \mathbb{R}^{T| \times |T|}$ is the transition parameter, indicating how possible a label will transfer to another. $T(c_{1:n})$ indicates all possible tag sequences for sentence $c_{1:n}$.

Also, we employ the Viterbi algorithm [Forney Jr, 1973] to decode the best tag sequence in polynomial time complexity.

3 A Feature-Enriched Neural Model for Joint S&T

The simple neural model presented above achieves good results on the joint S&T task. However, the simple neural model, who concatenates the embeddings of contextual characters as features, is not as strong as models based on the hand-crafted features. Thus, a simple shallow neural is insufficient to tackle with ambiguous cases which rely on more sophisticated feature combinations and long distance dependencies.

To deal with these issues, we propose a feature-enriched neural model for joint S&T task, which consists of three different types of neural layers, stacked one by one: (1) Convolutional layer; (2) Highway layer; (3) Recurrent layer. Figure 3 gives the illustration.
3.1 Convolutional Layer

The simple neural model is just to concatenate the embeddings of characters in a local context, which cannot simulate the carefully designed features in traditional models.

To better model the complex compositional features as conventional feature based models, we use convolution layer to separately model different n-gram features for each character. Thus the feature of each character is the concatenation of corresponding columns of all different feature map sets. Then we apply a k-max pooling layer to select the most significant signals.

Concretely, we model uni-gram, bi-gram, ..., Q-gram features by generating feature map sets $\hat{z}^1, \hat{z}^2, ..., \hat{z}^Q$ correspondingly. Formally, the q-gram feature map set $\hat{z}^q$ is:

$$\hat{z}^q = \tanh(W_{cov}^q \ast x_{i-\left\lfloor \frac{q-1}{2} \right\rfloor :i+\left\lfloor \frac{q+1}{2} \right\rfloor} + b), i \in [1, n], \quad (5)$$

where $W_{cov}^q \in \mathbb{R}^{d \times l_q}$ is the convolutional filter for q-gram feature map set, and $x_{i-\left\lfloor \frac{q-1}{2} \right\rfloor :i+\left\lfloor \frac{q+1}{2} \right\rfloor} \in \mathbb{R}^d$ is the concatenation of embeddings of characters $c_{i-\left\lfloor \frac{q-1}{2} \right\rfloor :i+\left\lfloor \frac{q+1}{2} \right\rfloor}$. Here, $l_q$ is the number of feature maps in q-gram feature map set and $b \in \mathbb{R}^d$ is a bias parameter. For marginal cases, we use wide convolution strategy, which means we receive the sequence in the same length as input by padding zeros to the input.

Then, we would represent the original sentence by concatenation operation as $z \in \mathbb{R}^{n \times \sum_q l_q}$:

$$z_i = \oplus_{q=1}^Q \hat{z}_i^q, \quad (6)$$

where operator $\oplus$ is the concatenation operation.

After taking the k-max pooling layer, the representation of original sentence would be $\hat{X} \in \mathbb{R}^{n \times d} = [\hat{x}_1, \hat{x}_2, ..., \hat{x}_n]^T$, where $\hat{x}_i$ is:

$$\hat{x}_i = k \max z_i, k = d. \quad (7)$$

Hence, after convolutional layer, we would represent the given input sequence $X \in \mathbb{R}^{n \times d} = [x_1, x_2, ..., x_n]^T$ as $\hat{X} = Cov(X)$.

3.2 Highway Layer

Highway layer [Srivastava et al., 2015] aims to keep gradient in very deep neural network. By introducing highway layer, we could simulate more complex compositional features by increasing the depth of our architecture. In addition, highway layer speeds up convergence speed and alleviates the problem of gradient vanishing.

As described above, we would represent the input sequence as $X = Cov(X)$ after the convolutional layer. By additionally adding the highway layer, the representation of the input sequence would be $\hat{X}$ as:

$$\hat{X} = Cov(X) \odot T(X) + X \odot C(X), \quad (8)$$

where operator $\odot$ indicates the element-wise multiplication operation. The $T(\cdot)$ is the transform gate and $C(\cdot)$ is the carry gate. We adopt a simple version, where we set $C(\cdot) = 1 - T(\cdot)$. Transform gate $T(\cdot)$ could be formalized as:

$$T(X) = \sigma(W_T^TX + b_T), \quad (9)$$

where $W_T \in \mathbb{R}^{d \times d}$ and $b_T \in \mathbb{R}^d$ are trainable parameters. Here $\sigma$ is the sigmoid function.

3.3 Recurrent Layer

In joint S&T task, it usually relies on long distance dependency and sophisticated features to disambiguate lots of cases. Thus, a simple shallow neural model is insufficient to capture long distance information.

Inspired by recent works using long short-term memory (LSTM) [Hochreiter and Schmidhuber, 1997] neural networks, we utilize LSTM to capture the long-term and short-term dependencies. LSTM is an extension of the recurrent neural network (RNN) [Elman, 1990], which aims to avoid the problems of gradient vanishing and explosion, and is very suitable to carry the long dependency information.

By further adding LSTM layer on the top of $\hat{X} \in \mathbb{R}^{n \times d} = [\hat{x}_1, \hat{x}_2, ..., \hat{x}_n]$, we would represent sentence $c_{1:n}$ as $H \in \mathbb{R}^{n \times h} = LSTM(\hat{X}) = [h_1, h_2, ..., h_n]$. Specifically, LSTM
layer introduces memory cell $c \in \mathbb{R}^h$ which controlled by input gate $i \in \mathbb{R}^h$, forget gate $f \in \mathbb{R}^h$ and output gate $o \in \mathbb{R}^h$. Thus, each output $h_i \in \mathbb{R}^h$ would be calculated as:

\[
\begin{pmatrix}
i_i \\
o_i \\
f_i \\
c_i \\
h_i
\end{pmatrix} = \begin{pmatrix}
\sigma \\
\sigma \\
\phi \\
\phi
\end{pmatrix} \left( W_g \begin{pmatrix}
\tilde{x}_i \\
h_{i-1} + b_g
\end{pmatrix} \right),
\]

\[
c_i = c_{i-1} \circ f_i + c_i \circ i_i,
\]

\[
h_i = o_i \circ \sigma(c_i),
\]

where $W_g \in \mathbb{R}^{4h \times (d + h)}$ and $b_g \in \mathbb{R}^h$ are trainable parameters. Here, the hyper-parameter $h$ is dimensionality of $i$, $o$, $f$, $c$ and $h$. $\sigma(\cdot)$ is sigmoid function and $\phi(\cdot)$ is $tanh$ function.

**BLSTM** We also employ the bi-directional LSTM (BLSTM) neural network. Specifically, each hidden state of BLSTM is formalized as:

\[
h_i = \overrightarrow{h}_i \oplus \overleftarrow{h}_i,
\]

where operator $\oplus$ indicates concatenation operation. Here, $\overrightarrow{h}_i$ and $\overleftarrow{h}_i$ are hidden states of forward and backward LSTMs respectively.

### 4 Training

We employ max-margin criterion [Taskar et al., 2005] which provides an alternative to probabilistic based methods by optimizing on the robustness of decision boundary directly.

In the decoding phase, if the predicted tag sequence for the $i$-th training sentence $t_{1:n_i}^{(i)}$ with the maximal score is $\hat{t}_{1:n_i}^{(i)}$, the goal of the max-margin criterion is to maximize the score of the gold tag sequence $t_{1:n_i}^{(i)} = \hat{t}_{1:n_i}^{(i)}$, with a margin to any other possible tag sequence $t_{1:n_i}^{(i)} \in \mathcal{T}(c_{1:n_i}^{(i)})$:

\[
\hat{t}_{1:n_i}^{(i)} = \arg \max_{t_{1:n_i}^{(i)} \in \mathcal{T}(c_{1:n_i}^{(i)})} r(t_{1:n_i}^{(i)}; \theta),
\]

\[
r(t_{1:n_i}^{(i)}; \theta) \equiv r(t_{1:n_i}^{(i)}; \theta) + \Delta(t_{1:n_i}^{(i)}; t_{1:n_i}^{(i)}),
\]

\[
\Delta(t_{1:n_i}^{(i)}; t_{1:n_i}^{(i)}) = \sum_{j=1}^{n_i} \eta \{ t_{j}^{(i)} \neq \hat{t}_{j}^{(i)} \},
\]

where $\Delta(t_{1:n_i}^{(i)}; t_{1:n_i}^{(i)})$ is the margin function and hyper-parameter $\eta$ is a discount parameter. Here, $\theta$ denotes all trainable parameters of our model.

Thus, the object is to minimize objective function $J(\theta)$ for $m$ training examples $(c_{1:n_i}^{(i)}, t_{1:n_i}^{(i)})_{i=1}^{m}$:

\[
J(\theta) = \frac{1}{m} \sum_{i=1}^{m} l_i(\theta) + \frac{\lambda}{2} \| \theta \|_2^2,
\]

\[
l_i(\theta) = \max_{t_{1:n_i}^{(i)} \in \mathcal{T}(c_{1:n_i}^{(i)})} \left( r(t_{1:n_i}^{(i)}; \theta) + \Delta(t_{1:n_i}^{(i)}; t_{1:n_i}^{(i)}) \right) - r(t_{1:n_i}^{(i)}; \theta).
\]

### 5 Experiments

#### 5.1 Datasets

We evaluate proposed architecture on five datasets: CTB, PKU, NCC, CTB-5, CTB-7. Table 1 gives the details of five datasets. We use the first 10% data of shuffled train set as development set for CTB, PKU, NCC and CTB-7 datasets.

- **CTB, PKU and NCC** datasets are from the POS tagging task of the Fourth International Chinese Language Processing Bakeoff [Jin and Chen, 2008].
- **CTB-5** dataset is the version of Penn Chinese Treebank 5.1, following the partition criterion of [Jin and Chen, 2008; Jiang et al., 2009; Sun and Wan, 2012].
- **CTB-7** dataset is the version of Penn Chinese Treebank 7.0. It consists of different sources of documents (newswire, magazine articles, broadcast news, broadcast conversations, newsgroups and weblogs). Since the web blogs are very different with news texts, we try to evaluate the robustness of our model by testing on web blogs and training on the rest of dataset.

#### 5.2 Hyper-parameters

Table 2 gives the details of hyper-parameter settings. Note that we set window size $k = 1$ which means we only take the current character embedding into account instead of using window slice approach. According to experiment results, we find it is a tradeoff between model performance and efficiency to only use \{ uni-gram, bi-gram, \ldots, 5-gram \} convolutional feature map sets. Besides, we set sizes of all feature map sets

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Splits</th>
<th>AVGw</th>
<th>N_sentence</th>
<th>DW</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTB</td>
<td>Train</td>
<td>27.4</td>
<td>23,444</td>
<td>42k</td>
</tr>
<tr>
<td></td>
<td>Test</td>
<td>28.8</td>
<td>2,079</td>
<td>10k</td>
</tr>
<tr>
<td>PKU</td>
<td>Train</td>
<td>16.7</td>
<td>66,691</td>
<td>55k</td>
</tr>
<tr>
<td></td>
<td>Test</td>
<td>24.3</td>
<td>6,424</td>
<td>18k</td>
</tr>
<tr>
<td>NCC</td>
<td>Train</td>
<td>28.4</td>
<td>18,869</td>
<td>45k</td>
</tr>
<tr>
<td></td>
<td>Test</td>
<td>28.5</td>
<td>3,595</td>
<td>18k</td>
</tr>
<tr>
<td>CTB-5</td>
<td>Train</td>
<td>27.3</td>
<td>18,068</td>
<td>37k</td>
</tr>
<tr>
<td></td>
<td>Dev</td>
<td>19.4</td>
<td>350</td>
<td>2k</td>
</tr>
<tr>
<td></td>
<td>Test</td>
<td>23.0</td>
<td>348</td>
<td>2k</td>
</tr>
<tr>
<td>CTB-7</td>
<td>Train</td>
<td>24.0</td>
<td>41,266</td>
<td>52k</td>
</tr>
<tr>
<td></td>
<td>Test</td>
<td>20.6</td>
<td>10,181</td>
<td>21k</td>
</tr>
</tbody>
</table>

Table 1: Details of five datasets. $DW$ is the dictionary of distinct words. $N_{sentence}$ indicates the number of sentences. $AVG_w$ is the average word number in a sentence.

<table>
<thead>
<tr>
<th>Window size</th>
<th>Character embedding size</th>
<th>Initial learning rate</th>
<th>Margin loss discount</th>
<th>LSTM dimensionality</th>
<th>Number of feature map sets</th>
<th>Size of each feature map set $f_i$</th>
<th>Batch size</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k = 1$</td>
<td></td>
<td>$d = 50$</td>
<td>$\alpha = 0.2$</td>
<td>$\eta = 0.2$</td>
<td>$\lambda = 10^{-4}$</td>
<td>$h = 100$</td>
<td>$Q = 5$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$f_i^Q = 100$</td>
<td>$20$</td>
</tr>
</tbody>
</table>

Table 2: Hyper-parameter settings.
By introducing convolutional layer and highway layer, we could further boost the performance which benefits from the feature modeling capability of convolutional layer and highway layer.

### 5.4 Comparison with Previous Works

We compare proposed model with several previous works on five datasets on joint S&T task. Experimental results are shown in Table 4.

Conditional random field (CRF) [Lafferty et al., 2001] is one of the most prevalent and widely used models for sequence labeling tasks. [Qiu et al., 2013] aims to boost the performance by exploiting datasets with different annotation types. Multilayer perceptron (MLP) is our implementation of [Zheng et al., 2013], a basic neural model for joint S&T task. [Zheng et al., 2013] is a neural model which only use one layer of shallow feed forward neural network in the encoding phase. Our model indicates the model with convolutional layer, pooling layer, highway layer, and BLSTM layer. “Pre-train” indicates the pre-trained character embeddings which are trained on corresponding train set of each dataset using word2vec toolkit [Mikolov et al., 2013].

### Result Discussion

Our model outperforms the previous neural model on joint S&T task and achieves the comparable performance with conventional hand-crafted feature based models. As shown in Table 4, compared to other previous methods, our model achieves the best performances on F1 scores (90.39, 90.16, 88.76, 85.31 on CTB, PKU, NCC and CTB-7 datasets respectively), and obtains comparable results on CTB5 dataset (93.19 on F1 score). As we know, the test set of CTB5 is very small so that previous work might over-fit on that dataset. In addition, according to the experimental results, we find that the performance benefits a lot from pre-trained character embeddings. Intuitively, pre-trained embeddings give a more reasonable initialization for the non-convex optimization problem with huge parameter space.

---

* [Zheng et al., 2013] only reported the results on CTB5 dataset for joint S&T task.
Besides, the proposed model is quite efficient. It only takes about half one hour per epoch using a small amount of memory (to train CTB) on a single GPU. Actually, it takes about ten hours to train our model (on CTB).

Experiments on CTB-7, whose train set and test set are on different domains, show the robustness of our model.

### 5.5 Case Study

We illustrate several cases from CTB-5 dataset. As shown in Table 5, our approach performs well on cases with ambiguations which rely on long distance dependency. For instance, conditional random field (CRF) model makes mistakes on words “改革” and “精简” since it is hard for CRF to disambiguate the POS tags without using the long distance (wider contextual) information.

### 6 Related Works

Recently, researches applied deep learning algorithms on various NLP tasks and achieved impressive results, such as chunking, POS tagging, named entity recognition for English [Collobert et al., 2011; Tsuboi, 2014; Labeau et al., 2015; Ma and Hovy, 2016; Santos and Zadrozy, 2014; Huang et al., 2015], and Chinese word segmentation and POS tagging for Chinese [Zheng et al., 2013; Pei et al., 2014; Chen et al., 2017]. These models learn features automatically which alleviate the efforts in feature engineering. However, joint S&T is a more difficult task than Chinese word segmentation and POS tagging since it has a larger decoding space and need more contextual information and long distance dependency [Zhang and Clark, 2008; Jiang et al., 2008; Kruengkrai et al., 2009; Zhang and Clark, 2010; Sun, 2011; Qian and Liu, 2012; Zheng et al., 2013; Qiu et al., 2013; Shen et al., 2014]. Therefore, we need a customized architecture to alleviate these problems. In this work, we propose a feature-enriched neural model for joint S&T task, and obtain great performance.

Besides, there are several similar neural models [Tsuboi, 2014; Labeau et al., 2015; Ma and Hovy, 2016; Santos and Zadrozy, 2014; Huang et al., 2015; Kim et al., 2015] . Instead of looking up word embedding table for each word in text, they tries to directly model English words by applying convolution layer on characters of words. Then they apply these word presentations to other tasks, such as POS tagging, name entity recognition, language modeling, etc. Unlike these models, we apply convolutional operation on sentence level, while they do within each word. Therefore they do not capture the features involving several words. Besides, we apply pooling operation along the feature size direction to get the most significant features.

### 7 Conclusions

In this paper, we propose a feature-enriched neural model for joint S&T task, which better models compositional features and utilizes long distance dependency. Experimental results show that our proposed model outperforms the previous neural model and achieves comparable results with previous sophisticated feature based approaches.

### Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. This work is partially funded by National Natural Science Foundation of China (No. 61532011 and 61672162), Shanghai Municipal Science and Technology Commission on (No. 16JC1420401).

### References


