Word-Context Character Embeddings for Chinese Word Segmentation

Hao Zhou∗
Nanjing University
& Toutiao AI Lab
zhouchao.nlp@bytedance.com

Zhenting Yu∗
Nanjing University
yuzt@nlp.nju.edu.cn

Yue Zhang
Singapore University
of Technology and Design
yue_zhang@sutd.edu.sg

Shujian Huang
Nanjing University
huangsj@nlp.nju.edu.cn

Xinyu Dai
Nanjing University
daixinyu@nju.edu.cn

Jiajun Chen
Nanjing University
chenjj@nlp.nju.edu.cn

Abstract

Neural parsers have benefited from automatically labeled data via dependency-context word embeddings. We investigate training character embeddings on a word-based context in a similar way, showing that the simple method significantly improves state-of-the-art neural word segmentation models, beating tri-training baselines for leveraging auto-segmented data.

1 Introduction

Neural network Chinese word segmentation (CWS) models (Zhang et al., 2016; Liu et al., 2016; Cai and Zhao, 2016) appeal for their strong ability of feature representation, employing unigram and bigram character embeddings as input features (Zheng et al., 2013; Pei et al., 2014; Ma and Hinrichs, 2015; Chen et al., 2015a). They give state-of-the-art performances. We investigate leveraging automatically segmented texts for enhancing their accuracies.

Such semi-supervised methods can be divided into two main categories. The first one is bootstrapping, which includes self-training and tri-training. The idea is to generate more training instances by automatically labeling large-scale data. Self-training (Yarowsky, 1995; McClosky et al., 2006; Huang et al., 2010; Liu and Zhang, 2012) labels additional data by using the base classifier itself, and tri-training (Zhou and Li, 2005; Li et al., 2014) uses two extra classifiers, taking the instances with the same labels for additional training data. A second semi-supervised learning method in NLP is knowledge distillation, which extracts knowledge from large-scale auto-labeled data as features.

Tri-training has been used in neural parsing, giving considerable improvements for both of dependency (Weiss et al., 2015) and constituent parsing (Vinyals et al., 2015; Choe and Charniak, 2016). Knowledge from auto-labeled data has also been used for parsing (Bansal et al., 2014; Melamud et al., 2016), where word embeddings are trained on automatic dependency tree context. Such knowledge has also been proved effective in conventional discrete CWS models, such as label distribution information (Wang et al., 2011; Zhang et al., 2013). However, it has not been investigated for neural CWS.

We propose word-context character embeddings (WCC), using segmentation label information in the pre-training of unigram and bigram character embeddings. The method packs the label distribution information into the embeddings, which could be regarded as a way for knowledge parameterization. Our idea follows Levy and Goldberg (2014), who use dependency contexts to train word embeddings. Additionally, motivated by co-training, we propose multi-view word-context character embeddings for cross-domain segmentation, which pre-trains two types of embedding for in-domain and out-of-domain data, respectively. In-domain embeddings are used for solving data sparseness, and out-of-domain embeddings are used for domain adaptation.

Our proposed model is simple, efficient and effective, giving average 1% accuracy improvement on in-domain data and 3.5% on out-of-domain data, respectively, significantly out-performing self-training and tri-training methods for leveraging auto-segmented data.

2 Baseline Segmentation Model

Chinese word segmentation can be regarded as a character sequence labeling task, where each char-
character in the sentence is assigned a segment label from left to right, including \{B, M, E, S\}, to indicate the segmentation (Xue, 2003; Low et al., 2005; Zhao et al., 2006). B, M, E represent the character is the beginning, middle or end of a multi-character word, respectively. S represents that the current character is a single character word.

Following Chen et al. (2015b), a standard bi-LSTM model (Graves, 2008) is used to assign segmentation label for each character. As shown in Figure 1, our model consists of a representation layer and a scoring layer. The representation layer utilizes a bi-LSTM to capture the context of each character in the sentence. Given a sentence \{w_1, w_2, w_3, \cdots, w_N\}, where \(w_i\) is the \(i^{th}\) character in the sentence, and \(N\) is the sentence length, we have a corresponding embedding \(e_{w_i}\) and \(e_{w_{i-1}w_i}\) for each character unigram \(w_i\) and character bigram \(w_{i-1}w_i\), respectively. A forward word representation \(e_i^f\) is calculated as follows:

\[
e_i^f = \text{concat}(e_{w_i}, e_{w_{i-1}w_i}), \\
= \tanh(W_1[e_{w_i}; e_{w_{i-1}w_i}])
\]

A backward representation \(e_i^b\) can be obtained in the same way. Then \(e_i^f\) and \(e_i^b\) are fed into forward and backward LSTM units at current position, obtaining the corresponding forward and backward LSTM representations \(r_i^{\text{lstm-}\text{f}}\) and \(r_i^{\text{lstm-}\text{b}}\), respectively.

In the scoring layer, we first obtain a linear combination of \(r_i^{\text{lstm-}\text{f}}\) and \(r_i^{\text{lstm-}\text{b}}\), which is the final representation at the \(i^{th}\) position.

\[
r_i = \text{concat}_2(r_i^{\text{lstm-}\text{f}}, r_i^{\text{lstm-}\text{b}}) \\
= \tanh(W_2[r_i^{\text{lstm-}\text{f}}; r_i^{\text{lstm-}\text{b}}])
\]

Given the representation \(r_i\), we use a scoring unit to score for each potential segment label. Given \(r_i\), the score of segment label \(M\) is:

\[
f_M^i = W_M h_i,
\]

where

\[
h = \text{concat}_3(r_i, e_M), \\
= \tanh(W_3[r_i; e_M])
\]

\(W_M\) is the score matrix for label \(M\), and \(e_M\) is the label embedding for label \(M\).

### 3 Word-Context Character Embeddings

Our model structure is a derivation from the skip-gram model (Mikolov et al., 2013), similar to Levy and Goldberg (2014). Given a sentence with length \(n\): \{\(w_1, w_2, w_3, \cdots, w_N\)\} and its corresponding segment labels: \{\(l_1, l_2, l_3, \cdots, l_n\)\}, the pre-training context of current character \(w_i\) is the around characters in the windows with size \(c\), together with their corresponding segment labels (Figure 2). Characters \(w_i\) and labels \(l_i\) in the context are represented by vectors \(e_{w_i}^c \in \mathbb{R}^d\) and \(e_{l_i}^c \in \mathbb{R}^d\), respectively, where \(d\) is the embedding dimensionality.

The word-context embedding of character \(w_i\) is represented as \(e_{w_i} \in \mathbb{R}^d\), which is trained by predicting the surrounding context representations \(e_{w_i}'\)
and $e_{l_i}^c$, parameterizing the labeled segmentation information in the embedding parameters. To capture order information (Ling et al., 2015), we use different embedding matrices for context embedding in different context positions, training different embeddings for the same word when they reside on different locations as the context word. In particular, our context window size is five. As a result, each word has four different versions of $e^c$, namely $e_{l-1}^c$, $e_{l-2}^c$, $e_{l+1}^c$, and $e_{l+2}^c$, each taking a distinct embedding matrix. Given the context window $[w_{l-2}, w_{l-1}, w, w_{l+1}, w_{l+2}]$, $w_{l-1}$ is the left first context word of the focus word $w$, $e_{l-1,w}^c$ will be selected from embedding matrix $E_{l-1}$, and $w_{l+1}$ is the right first word of $w$, $e_{l+1,w}^c$ will be selected from embedding matrix $E_{l+1}$.

Note that each character has two types of embeddings, where $e_{w_i}$ is the embedding form of $w_i$ when $w_i$ is the focus word, and $e_{c,w_i}$ is the embedding form of $w_i$ when $w_i$ is used as a surrounding context word. We do not have $e_{l_i}$ because $l_i$ only acts as the surrounding context. After pre-training, $e_{w_i}$ will be used as the WCC embeddings.

The objective of our model is to maximize the average log probability of the context:

$$\frac{1}{T} \sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j}|w_t) + \log p(t_{t+j}|w_t)$$

Negative sampling (Mikolov et al., 2013) is used, where $\log p(w_{t+j}|w_t)$ and $\log p(t_{t+j}|w_t)$ are computed as:

$$p(w_{t+j}|w_t) = \log \sigma(e_{w_{t+j}}^c \top e_{w_t})$$
$$+ \sum_{i=1}^{k} E_{w_i \sim P_n(w)} \log \sigma(-e_{w_i}^c \top e_{w_t})$$

and

$$p(t_{t+j}|w_t) = \log \sigma(e_{t_{t+j}}^c \top e_{w_t})$$
$$+ \sum_{i=1}^{k} E_{l_i \sim P_n(l)} \log \sigma(-e_{l_i}^c \top e_{w_t})$$

respectively, where $P_n(w)$ and $P_n(l)$ is the noise distributions and $k$ is the size of negative samples for each data sample.

Bigram embeddings are trained in the same way as unigram character embeddings. For out-of-domain segmentation, we pre-train two embeddings for each token, extracting knowledge from the two domains, respectively.

---

4 Experiments

4.1 Set-up

We perform experiments on three standard datasets for Chinese word segmentation: PKU and MSR from the second SIGHAN bakeoff shared task, and Chinese Treebank 6.0 (CTB6). For PKU and MSR, 10% of the training data are randomly selected as development data. We follow Zhang et al. (2016) to split the CTB6 corpus into training, development and testing sections. For evaluating cross-domain performance, we also experiment on Chinese novel data. Following Zhang et al. (2014), the training set of CTB5 is selected for training, and the manually annotated sentences of free Internet novel ’Zhuxian’ (ZX) are selected as the development and test data (Liu and Zhang, 2012)\(^1\).

Chinese Gigaword (LDC2011T13, 4M) is used for in-domain unlabeled data. For out-of-domain data, 20K raw sentences of Zhuxian is used. We take self-training and tri-training as baselines, which also use large-scale auto-segmented data. For self-training, skip-gram pre-training and word-context character embedding, unlabeled corpus is segmented automatically by our baseline model. For tri-training, we additionally use the ZPar (Zhang and Clark, 2007) and ICTCLAS\(^2\) as our base classifiers.

We use F1 to evaluate segmentation accuracy. The recalls of in-vocabulary (IV) and out-of-vocabulary (OOV) are also measured.

4.2 Hyper-Parameters

The hyper-parameters used in this work are listed in Table 1. The values are selected according

\(^1\)http://zhangmeishan.github.io/acl14mszhang.zip
\(^2\)http://ictclas.nlpir.org/
Table 1: Hyper-parameters.

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>unigram dimension</td>
<td>50</td>
</tr>
<tr>
<td>bigram dimension</td>
<td>50</td>
</tr>
<tr>
<td>label embedding dimention</td>
<td>32</td>
</tr>
<tr>
<td>LSTM hidden size</td>
<td>100</td>
</tr>
<tr>
<td>LSTM input size</td>
<td>100</td>
</tr>
<tr>
<td>learning rate</td>
<td>0.1</td>
</tr>
<tr>
<td>window size</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 2: Comparisons between greedy and CRF segmentation. Speed: tokens per millisecond.

<table>
<thead>
<tr>
<th>System</th>
<th>CTB6</th>
<th>PKU</th>
<th>MSR</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greedy</td>
<td>94.9</td>
<td>95.0</td>
<td>97.2</td>
<td>14.7</td>
</tr>
<tr>
<td>CRF</td>
<td>95.0</td>
<td>95.1</td>
<td>97.2</td>
<td>3.6</td>
</tr>
</tbody>
</table>

Table 3: Results of self-training and tri-training on CTB6 with varying scaled training data.

<table>
<thead>
<tr>
<th>Type</th>
<th>System</th>
<th>CTB6</th>
<th>PKU</th>
<th>MSR</th>
</tr>
</thead>
<tbody>
<tr>
<td>non-nn</td>
<td>Iseng et al. (2005)</td>
<td>-</td>
<td>95.0</td>
<td>96.4</td>
</tr>
<tr>
<td></td>
<td>Sun et al. (2009)</td>
<td>-</td>
<td>95.2</td>
<td>97.3</td>
</tr>
<tr>
<td></td>
<td>Wang et al. (2011)</td>
<td>95.8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Zhang et al. (2013)</td>
<td>-</td>
<td>96.1</td>
<td>97.5</td>
</tr>
<tr>
<td>mn</td>
<td>Zheng et al. (2013)</td>
<td>-</td>
<td>92.4</td>
<td>93.3</td>
</tr>
<tr>
<td></td>
<td>Pei et al. (2014)</td>
<td>95.2</td>
<td>97.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Kong et al. (2015)</td>
<td>90.6</td>
<td>90.7</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ma and Hinrichs (2015)</td>
<td>-</td>
<td>95.1</td>
<td>96.6</td>
</tr>
<tr>
<td></td>
<td>Zhang et al. (2016)</td>
<td>95.8</td>
<td>96.1</td>
<td>96.3</td>
</tr>
<tr>
<td></td>
<td>Cai and Zhao (2016)</td>
<td>95.5</td>
<td>95.7</td>
<td>97.6</td>
</tr>
<tr>
<td>comb</td>
<td>Zhang et al. (2016)</td>
<td>96.0</td>
<td>95.7</td>
<td>97.7</td>
</tr>
<tr>
<td></td>
<td>baseline</td>
<td>94.9</td>
<td>95.0</td>
<td>97.2</td>
</tr>
<tr>
<td></td>
<td>+ self-training</td>
<td>95.0</td>
<td>94.8</td>
<td>97.0</td>
</tr>
<tr>
<td></td>
<td>+ tri-training</td>
<td>95.5</td>
<td>95.5</td>
<td>97.4</td>
</tr>
<tr>
<td></td>
<td>+ skip-gram embeddings</td>
<td>95.3</td>
<td>95.5</td>
<td>97.4</td>
</tr>
<tr>
<td></td>
<td>+ WCC embeddings</td>
<td>96.2</td>
<td>96.0</td>
<td>97.8</td>
</tr>
</tbody>
</table>

Table 4: Comparison with other models.

<table>
<thead>
<tr>
<th>Type</th>
<th>System</th>
<th>CTB6</th>
<th>PKU</th>
<th>MSR</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>baseline</td>
<td>94.9</td>
<td>95.0</td>
<td>97.2</td>
</tr>
</tbody>
</table>

4.3 Utilizing Varying-Scale Data

The results of self-training and tri-training with varying-scale training data are list in Table 3, where +4X means adding 4 times the size of supervised training data into the training set. We find that self-training does not work well, and tri-training with 16X gives a 0.5% accuracy improvement. We adopt this setting for our baseline in the remaining experiments.

We also try to choose more effective examples for self-training and tri-training, by selecting training instances according to the base segmentation model score. However, the segmentation performances do not get improved. A possible reason is that the training instances with higher confidence are always shorter than the original sampled sentences, which may not be very helpful for semi-supervised segmentation.

4.4 In-Domain Results

As shown in Table 4, pre-training with conventional skip-gram embeddings gives only small improvements, which is consistent as findings of previous work (Chen et al., 2015a; Ma and Hinrichs, 2015; Cai and Zhao, 2016). Segmentation with self-training even shows accuracy drops on PKU and MSR. We speculate that the self-training by the neural CWS baseline is sensitive to the segmentation errors of the auto-labeled data. On average, our method obtains an absolute 1% accuracy improvement over the baseline, outperforming other semi-supervised method significantly.

We compare our model with other state-of-the-art segmentation models, which are grouped into 3 classes, namely traditional segmentation models (non-NN), neural segmentation models (nn), and the combination of both neural and traditional discrete features (comb). Our simple model gives top accuracies compared with related work. Liu et al. (2016), Cai and Zhao (2016) and Zhang et al. (2016) propose to incorporate word embedding features in the neural CWS, pre-training the word embeddings in the large-scale labeled data. Different to them, we employ a simpler character level model containing word information, yet obtaining higher F1 scores.
4.5 Out-of-Domain Results

We test out-of-domain performance of our model on the ZX dataset. We also use the multi-view word-context character embeddings (WCC) for cross domain segmentation, which uses two types of embeddings by simple vector concatenation. One type of embeddings is pre-trained on in-domain data, and the other type is pre-trained on out-of-domain data. In such case, the multi-view embeddings includes cross-domain information, which may enhance the cross-domain segmentation performance (Mou et al., 2016).

As shown in Table 5, using word-context character (WCC) embeddings and multi-view word-context character embeddings both give significantly higher accuracy improvements compared with other semi-supervised methods. Additionally, we find that multi-view WCC embeddings give an extra 1% F1 score improvement over WCC embeddings. Our proposed model also significantly improves the OOV recall (ROOV) and IV recall (RIV). By studying the cases of segmented output (Figure 3), we find that our model can recognize OOV words such as ‘鬼王’, ‘七星剑’ and the IV word ‘器重’, which are incorrectly labeled by the baseline. This confirms that our proposed model is helpful for the data sparseness problem on closed domain and domain adaptation on across domain.

We also list the results of Zhang et al. (2014) and Liu et al. (2014) on this dataset. Liu et al. (2014) obtains better out-of-domain performance than our model. However, their results cannot be compared directly with ours because they use partial labeled URL link data from Chinese Wikipedia data for training.

5 Conclusion

We proposed word-context character embeddings for semi-supervised neural CWS, which makes the segmentation model more accurate on in-domain data, and more robust on the out-of-domain data. Our segmentation model is simple yet effective, achieving state-of-the-art segmentation accuracies on standard benchmarks. It can also be useful for other NLP tasks with small labeled training data, but a large unlabeled data. Our code could be downloaded at https://github.com/zhouh/WCC-Segmentation.

Acknowledgment

We would like to thank the anonymous reviewers for their insightful comments. We also thank Ji Ma and Meishan Zhang for their helpful discussions. This work was partially founded by the Natural Science Foundation of China (61672277, 71503124) and the China National 973 project 2014CB340301.

Table 5: Results on the out-of-domain data. Models with † do not use large-scale data, models with ‡ use in-domain large-scale data, and models with ⚫ use both in-domain, and out-of-domain large-scale data.

<table>
<thead>
<tr>
<th>System</th>
<th>F1</th>
<th>ROOV</th>
<th>RIV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zhang et al. (2014b)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>baseline</td>
<td>87.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+ self-training</td>
<td>88.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Liu et al. (2014)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>baseline</td>
<td>87.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+Chinese Wikipedia</td>
<td>90.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ours</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>baseline</td>
<td>86.6</td>
<td>60.8</td>
<td>91.7</td>
</tr>
<tr>
<td>+skip-gram</td>
<td>87.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+self-training</td>
<td>87.8</td>
<td>70.3</td>
<td>91.5</td>
</tr>
<tr>
<td>+tri-training</td>
<td>88.1</td>
<td>68.1</td>
<td>91.5</td>
</tr>
<tr>
<td>+WCC embeddings</td>
<td>89.7</td>
<td>70.4</td>
<td>93.7</td>
</tr>
<tr>
<td>+ multi-view WCC embeddings</td>
<td>90.1</td>
<td>74.1</td>
<td>93.3</td>
</tr>
</tbody>
</table>

References


