Improving Cross-Domain Chinese Word Segmentation with Word Embeddings

Yuxiao Ye[1]*, **Yue Zhang**[2*†], **Weikang Li**[3], **Likun Qiu**[1†], **Jian Sun**[1]
[1]Alibaba Group, [2]Peking University, [3]Westlake University
{yuxiao.yyx, likun.qlk, jian.sun}@alibaba-inc.com
yue.zhang@wias.org.cn
wavejkd@pku.edu.cn

Abstract

Cross-domain Chinese Word Segmentation (CWS) remains a challenge despite recent progress in neural-based CWS. The limited amount of annotated data in the target domain has been the key obstacle to a satisfactory performance. In this paper, we propose a semi-supervised word-based approach to improving cross-domain CWS given a baseline segmenter. Particularly, our model only deploys word embeddings trained on raw text in the target domain, discarding complex hand-crafted features and domain-specific dictionaries. Innovative subsampling and negative sampling methods are proposed to derive word embeddings optimized for CWS. We conduct experiments on five datasets in special domains, covering domains in novels, medicine, and patent. Results show that our model can obviously improve cross-domain CWS, especially in the segmentation of domain-specific noun entities. The word F-measure increases by over 3.0% on four datasets, outperforming state-of-the-art semi-supervised and unsupervised cross-domain CWS approaches with a large margin. We make our code and data available on Github.

1 Introduction

Chinese Word Segmentation (CWS) is the first step for many Chinese Natural Language Processing (NLP) tasks (Cai and Zhao, 2016; Zhao et al., 2017). Approaches to CWS could be categorized into two categories: character-based and word-based. The former treats CWS as a sequence labeling problem, labeling each character in a sequence with *B/I/E/S* (*Beginning, Internal, End, Single*) labels (Tseng et al., 2005). Traditional character-based approaches often use Conditional Random Fields (CRF) models to label sequences, with complex hand-crafted discrete features (Peng et al., 2004; Tseng et al., 2005). Unlike character based CWS, word-based CWS operates on a word-level, directly exploiting word-level features. Typical CRF models are replaced with semi-CRF models, in which labels are assigned to subsequences instead of characters (Sarawagi and Cohen, 2005; Liu et al., 2014). Transition-based approaches have also been used to exploit larger feature contexts (Zhang and Clark, 2007). More recent approaches exploit neural networks including Recurrent Neural Networks (RNN) to replace hand-crafted discrete features with real-valued features (Cai and Zhao, 2016; Chen et al., 2015, 2017).

Existing studies have achieved satisfactory results for in-domain CWS, with F-scores over 96.0% in the newspaper domain (Chen et al., 2017). Nevertheless, cross-domain CWS remains a big challenge (Liu et al., 2014; Liu and Zhang, 2012). The main reason is the lack of annotated data in the target domain, which makes supervised approaches less useful. To tackle this problem, some unsupervised and semi-supervised approaches have been proposed. One way is to exploit complex features including character types, lexical features and accessor varieties (Wu et al., 2014), which requires much efforts on feature engineering. Another way is to deploy machine learning algorithms including self-training and model ensemble (Gao and Stephan, 2010; Liu and Zhang, 2012; Qiu and Zhang, 2015), which is time-consuming and inefficient.

In this paper, we investigate a different approach to deploying unsupervised data for cross-domain CWS, in order to completely break free from the reliance on manual annotation, complex feature engineering, and even parametric training to some extent. We propose a Word-Embedding-

* Yuxiao Ye and Yue Zhang contributed equally to this work.
† Corresponding authors.

Proceedings of NAACL-HLT 2019, pages 2726–2735
Minneapolis, Minnesota, June 2 - June 7, 2019.

Based CWS (WEB-CWS) model, which aims to improve the performance of an existing baseline segmenter in cross-domain CWS. WEB-CWS is a conceptually simple word-based model, using word embeddings, which are expected to carry semantic and syntax information (Mitchell and Lapata, 2010), as the only input of a non-parametric word segmentor. The basic intuition is that embeddings of words within a same context window should be close to each other (Goldberg and Levy, 2014; Mikolov et al., 2013). If a sequence is incorrectly segmented, those incorrectly segmented words are likely to be semantically and syntactically inconsistent with their surrounding words. Consequently, the embedding of an incorrectly segmented word should be far away from embeddings of its surrounding words.

Based on the hypothesis above, we propose WEB-CWS. Word embeddings are first derived with a CWS-oriented word embedding model with innovative subsampling and negative sampling methods. A word-embedding-based decoder is then used for segmentation, with cosine similarities among word embeddings as the metric for probability calculation. WEB-CWS is a *semi-supervised* model, because it only uses word embeddings trained on raw text in the target domain, which is first automatically segmented by the baseline segmenter. The model is also *cross-domain* in the sense that it can improve the performance of the baseline segmenter, when the source text for training the baseline segmenter and the target text to be segmented are in different domains.

The main contributions of this paper include:

- To our knowledge, we are the first to directly use word embeddings for CWS, without any neural structures, which makes our model conceptually simpler and run faster.
- We have proposed novel sampling methods to make the embeddings optimized for CWS, which has never been used for embedding training.
- Our model can be used on top of any existing CWS models to improve their performances, without the need to re-train those models with annotated domain specific data.
- On four datasets in different special domains, our model improves the word F-measure by more than 3.0%, compared with the state-of-the-art baseline segmenter. We release our code and data on Github[1].

2 Related Work

Our work is related with existing research on word-based CWS, cross-domain CWS, and embedding-based CWS.

2.1 Word-Based CWS

Instead of labeling a sequence character-wise, word-based CWS tries to pick the most probable segmentation of a sequence. Zhang and Clark (2007) design a statistical method for word-based CWS, extracting word-level features directly from segmented text. The perceptron algorithm (Collins, 2002) is used for training and beam-search is used for decoding. Cai and Zhao (2016) use Gated Combination Neural Networks and LSTM to present both character sequences and partially segmented word sequences, combining word scores and link scores for segmentation. Our work is in line with their work in directly using word information for CWS. In contrast, our method is conceptually simpler by directly using word embeddings. In addition, our work aims at domain-adaptation, rather than training from scratch.

2.2 Cross-Domain CWS

Supervised, semi-supervised and unsupervised approaches have been proposed for domain adaptation for CWS. Chen et al. (2017) use an Adversarial Network to learn shared knowledge for different segmentation criteria and domains. This approach requires annotated data in the target domain.

However, one challenge for cross-domain CWS is the lack of such annotated data. Liu and Zhang (2012) propose an unsupervised model, in which they use features derived from character clustering, together with a self-training algorithm to jointly model CWS and POS-tagging. This approach is highly time-consuming (Qiu and Zhang, 2015). Another challenge is the segmentation of domain-specific noun entities. In a task of segmenting Chinese novels, Qiu and Zhang (2015) design a double-propagation algorithm with complex feature templates to iteratively extract noun entities and their context, to improve segmentation

[1] https://github.com/vatile/CWS-NAACL2019

performance. This approach still relies heavily on feature templates. Similarly, our model does not require any annotated target data. In contrast to their work, our model is efficient and feature-free.

2.3 CWS Using Embeddings

There are CWS models deploying embeddings. Ma and Hinrichs (2015) and Deng and Sun (2018) propose embedding matching CWS models, in which embeddings of characters in a sequence are compared with high dimensional representations of CWS-specific actions (e.g., separation and combination) or CWS-specific labels (e.g., *B/M/E/S*). Then each character is labeled according to the similarity between its embedding and the high dimensional representation.

Particularly, Zhou et al. (2017) propose to use character embeddings trained on a word-based context to improve the performance of existing neural CWS models, which is similar to our approach in terms of making use of CWS-oriented word embeddings derived with automatically segmented raw corpus. However, in their work, when doing cross-domain CWS, word embeddings are fed into the baseline neural model trained on a large annotated general corpus, with annotated special domain data as the development set. In our model, on the contrary, word embeddings are used directly for CWS with a non-parametric decoder, which does not require to re-construct the baseline model, and annotation is not required at all for the special domain data.

3 Word-Embedding-Based CWS

The overall architecture of our method is shown in Figure 1. Given a baseline segmenter and a target domain raw corpus T, we obtain an automatically segmented corpus $T^{'}$ by applying the baseline segmenter to the target corpus. We then execute our CWS-oriented word embedding model (Section 3.1) on $T^{'}$ to derive a set of word embeddings E. In addition, all tokens from $T^{'}$ are collected as a target domain dictionary D. Finally, E and D are used to re-segment T with our word-embedding-based segmenter (Section 3.2).

3.1 CWS-Oriented Word Embedding Model

We use a CWS-oriented model modified from the Skip-gram model (Mikolov et al., 2013) to derive word embeddings. A typical Skip-gram model using negative sampling tries to maximize the following objective (Mikolov et al., 2013):

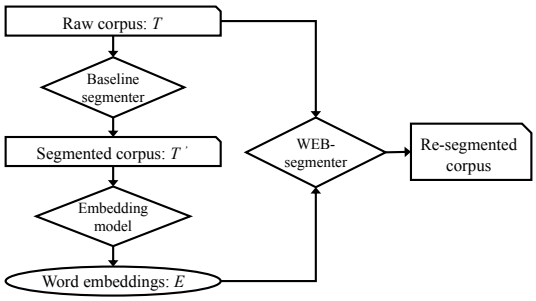

Figure 1: The pipeline of WEB-CWS.

$$\sum_{(w,c)\in P} \log \sigma(v_w \cdot v_c^\top) + \sum_{(w,c^{'})\in N} \log \sigma(-v_w \cdot v_{c^{'}}^\top) \quad (1)$$

with P being the set of positive samples (w, c) consisting of a target word w and a context word c, N being the set of negative samples $(w, c^{'})$ consisting of a target word w and a word $c^{'}$ drawn randomly from a noise distribution $P_n(w)$, v_w being the word embedding of w, and σ being the sigmoid activation function.

Subsampling is applied when choosing the target word w to reduce training time and to improve the quality of embeddings of rare words (Mikolov et al., 2013). For a natural language, the frequency distribution of all words is expected to obey Zipf's law: a word's frequency is inversely proportional to its rank in the frequency table (Newman, 2005). This highly biased distribution makes the training of the Skip-gram model inefficient, in that very frequent words can make a large portion of training samples, but their embeddings may not change much after being seen for a certain time (Mikolov et al., 2013). Therefore, a subsampling method is used by Mikolov et al. (2013), with the probability for a word w being sampled as:

$$p_{sub}(w) = min(1, \sqrt{\frac{\epsilon}{f(w)}}) \quad (2)$$

where ϵ is an arbitrarily chosen threshold, and $f(w)$ is the frequency of w.

Since word embeddings in our model are used for segmentation, we cannot directly use the training objective in Mikolov et al. (2013), which is designed for language modeling. To make the training objective more consistent with the goal of CWS, we modify the negative sampling Skip-gram model in various ways, including adding

CWS-oriented negative samples, changing the method for subsampling multi-character words, normalizing the dot product of embeddings, and smoothing the weights of positive and negative samples in training.

3.1.1 Context Negative Sampling

When training a typical Skip-gram model, a target word and a word within its context window are taken together as a positive sample (Mikolov et al., 2013). From the perspective of CWS, it can be perceived as teaching the model how to correctly segment a sequence. From this perspective, we develop a method to generate negative samples from a word's context (i.e., words within the context window), in order to tell the model what the incorrect segmentations of a sequence are.

Given a target word w and its context C, and SL/SR as the sequence of characters on the left/right of w within C, the proposed context negative sampling method generates negative samples in the following way: for any substring $s^{'}$ of SL and SR, if $s^{'}$ is in the dictionary D but not in C, (w, $s^{'}$) will be generated as a negative sample.

3.1.2 In-Word Negative Sampling

Another way to generate negative samples concerning CWS is to split multi-character words and combine its substrings as negative samples. For instance, given a multi-character target word $w = c_1c_2c_3$, supposing that all its substrings are in D, the proposed in-word negative sampling method will then generate the following negative samples: (c_1, c_2), (c_1, c_3), (c_2, c_3), (c_1c_2, c_3) and (c_1, c_2c_3). By doing so, our model is expected to learn not to split those multi-character words when segmenting.

3.1.3 Subsampling Multi-Character Words

In the Chinese language, there are some frequent multi-character words consisting of substrings which are also very frequent words themselves. For example, the Chinese word 'dànshì (but)' can be decomposed into two words 'dàn (but)' and 'shì (be)'. Due to the nature of the Skip-gram model, embeddings of frequent words are relatively close to each other since they co-occur with other words more frequently. This nature makes our model inclined to split such multi-character words when segmenting. Although the in-word negative sampling method proposed above is expected to prevent our model from incorrectly splitting multi-character words, we still want our model to pay more attention to the segmentation of such words. As a result, for subsampling, we will not discard a multi-character word w if:

$$p_{sub}(w) < \frac{\mu}{N} \sum_{w_{sub} \in S \cap D} p_{sub}(w_{sub}) \quad (3)$$

where S is the set of substrings of w, and N is the size of all $w_{sub} \in S \cap D$, which is smoothed by a threshold μ (which is empirically set to 0.5 in our model). By doing so, we can keep those multi-character words whose substrings are more frequent words themselves. Our model is thus expected to learn better how to segment such words through samples generated with in-word negative sampling.

3.1.4 Dot Product Normalization

In the original Skip-gram model, the dot product of embeddings of two words is directly used as the input for the sigmoid layer (Mikolov et al., 2013). To make word embeddings derived from the CWS-oriented word embedding model more consistent with the metric used for segmentation as described in Section 3.2.2, we modify the training objective described in Equation (1) as follows:

$$\sum_{(w,c) \in P} \log \sigma(||v_w \cdot v_c^\top||) + \sum_{(w,c') \in N} \log \sigma(-||v_w \cdot v_{c'}^\top||) \quad (4)$$

3.1.5 Smoothing Class Weights

For any target word, when training the CWS-oriented word embedding model, only one positive sample but many negative samples are generated. To balance the influence of positive and negative samples, a different weight is assigned to each class as follows:

$$class\_weight = \begin{cases} 1.0 & , positive \\ (\frac{N_{pos}}{N_{neg}} + \eta)/(1+\eta) & , negative \end{cases} \quad (5)$$

where N_{pos} and N_{pos} are the amount of positive and negative samples respectively, and η is a smooth factor. The smooth factor can prevent the weight of negative samples being too low when negative samples are much more than positive samples.

3.2 Word-Embedding-Based Segmentater

In WEB-CWS, we formalize the process of segmenting a sequence as a problem of hypotheses-based Viterbi decoding (Forney, 1973): given a sequence, generating segmentation hypotheses character-wise from the first to the last character, and then searching for the optimal path according to the predefined metric of probability.

3.2.1 Hypothesis Generation

Given a sentence consisting of n characters $S =< c_0 > c_1 c_2 ... c_n < c_{n+1} >$ (c_0 and c_{n+1} are markers of the beginning/end of a sentence), we generate segmentation hypotheses character-wise from c_0 to c_{n+1}. At each time step t, a hypothesis h_t is defined as:

$$h_t = \begin{cases} SEG_t : [w_{0\{c_0\}} w_{1\{c_1 ... c_g\}} ... w_{m\{c_j ... c_k\}}] \\ BUF_t : [c_{k+1} ... c_t] \\ M_t = m + 1 \end{cases} \tag{6}$$

which includes a partial segmentation container SEG_t, a buffer container BUF_t, and the number of segmented words M_t. In h_t, characters $c_0 c_1 ... c_k$ are segmented into words $w_0 w_1 ... w_m$ stored in SEG_t; characters $c_{k+1} .. c_t$ remain unsegmented and are stored in BUF_t. For the initial hypothesis h_0 and the final hypothesis h_{n+1}, the buffer container will be empty.

Given a character c_{t+1} and a hypothesis h_t, h_{t+1} can be generated in two ways, by either appending c_{t+1} to BUF_t ($t \neq n$), or first moving the sequence in BUF_t into SEG_t as a new word, and then appending c_{t+1} to BUF_t ($t \neq n$). In the former case:

$$h_{t+1} = \begin{cases} SEG_t : [w_{0\{c_0\}} ... w_{m\{c_j ... c_k\}}] \\ BUF_t : [c_{k+1} ... c_t c_{t+1}] \\ M_t = m + 1 \end{cases} \tag{7}$$

In the latter case:

$$h_{t+1} = \begin{cases} SEG_t = w_{0\{c_0\}} ... w_{m\{c_j ... c_k\}} w_{m+1\{c_{k+1} ... c_t\}} \\ BUF_t = c_{t+1} \\ M_t = m + 2 \end{cases} \tag{8}$$

Particularly, when generating the hypothesis for c_{n+1} (the end of sentence marker), the sequence in BUF_n has to be moved into SEG_n, and c_{n+1} also needs to be moved into SEG_n as a new word.

If a word w is not in the dictionary D, we cannot get its embedding. As a result, any hypothesis containing w in the segmentation container will be discarded. Moreover, to reduce search space, once the size of the sequence in a buffer container reaches a threshold m (i.e., the maximum word length), this sequence will be moved into the segmentation container when generating hypotheses at the next time step.

3.2.2 Probability Calculation

The log probability of a hypothesis h_t is defined as:

$$\log p(h_t) = \begin{cases} 0, & t = 0 \\ \frac{1}{M_t - 1} \log \prod_{i=1}^{M_t - 1} p(w_i | SEG_t, f), & t \neq 0 \end{cases} \tag{9}$$

where f is the window size (e.g., if $f = 2$, $p(w_i | SEG_t, f)$ will be decided by w_{i-1} and w_{i-2}).

In WEB-CWS, we use cosine similarity between embeddings of two words as the metric of probability. Given a hypothesis h_t, $p(w_i | SEG_t, f)$ is calculated as follows:

$$p(w_i | SEG_t, f) = \begin{cases} e^0, & i = 0 \\ e^{\frac{1}{min(f,i)} \sum_{j=1}^{min(f,i)} \cos(v_{w_i}, v_{w_{i-j}})}, & i \neq 0 \end{cases} \tag{10}$$

where $\cos(v_{w_i}, v_{w_{i-j}})$ refers to the cosine similarity between embeddings of w_i and w_{i-j}.

Given a hypothesis h_t, the log probability of h_{t+1} can be dynamically computed as:

$$\log p(h_{t+1}) = \frac{1}{M_{t+1} - 1} ((M_t - 1) \log p(h_t) + \sum_{w_i \in \{SEG_{t+1} - SEG_t\}} \log p(w_i | SEG_{t+1}, f)) \tag{11}$$

3.2.3 Dynamic Beam-Size and Maximum Word Length

Theoretically, a sequence of length n can have at most 2^{n-1} possible segmentations. By discarding hypotheses containing out-of-vocabulary (OOV) words and setting the maximum word length, the search space can be significantly reduced. The very limited search space makes dynamically deciding the beam-size and the maximum word length possible.

Given an initial beam-size k, at each time step t, the segmenter will only keep at most top k hypothesis sorted by log probabilities in a descending order. Some hypotheses will also be discarded due to OOV words and the maximum word length limit. As a result, it is sometimes possible for a sequence to have no hypothesis at all after some time steps. Once it happens, the segmenter will increase the beam-size by 10 and the maximum word length by 1, and then re-generate hypothesis from the beginning, till at least one hypothesis is generated at the final time step.

Dynamic beam-size and maximum word length ensure that for each sequence, at least one segmentation (the one given by the baseline segmenter) will be generated as the final segmentation result. This mechanism can guarantee the efficiency and reliability of the segmenter at the same time.

3.2.4 Similarity Score Look-up

To improve the decoding speed, we pre-calculate the cosine similarity of all word pairs co-occurring at least once in the automatically segmented corpus, and store them in a file (which only consists of millions of word pairs for a Chinese novel). In doing so, when decoding, for most word pairs, we only need to look up to this file for similarity scores, which can significantly improve the decoding speed. According to later experiments, this look-up strategy can cover about 92% of the similarity calculation needed for decoding.

4 Experiments

We conduct experiments on various datasets in different domains to thoroughly evaluate the performance of our model.

4.1 Experimental Setup

4.1.1 Datasets

We evaluate our model in terms of cross-domain CWS on five datasets, including three Chinese novel datasets (Qiu and Zhang, 2015): DL (*DouLuoDaLu*), FR (*FanRenXiuXianZhuan*) and ZX (*ZhuXian*), and two CWS datasets in special domains (Qiu et al., 2015): DM (dermatology) and PT (patent). We use the standard split for all datasets as they are published. Raw test data is also included for deriving word embeddings.

Statistics of these datasets are shown in Table 1. Since there are no gold segmentation of full novels for three Chinese novel datasets, their statistics are based on the segmentation given by the baseline segmenter.

Dataset	Sentence (K)		Token (K)		Character (K)	
	Full	Eval	Full	Eval	Full	Eval
DL	40	1	1,982	32	2,867	47
FR	148	1	5,004	17	7,126	25
ZX	59	1	2,131	21	3,006	31
DM	32	1	709	17	1,150	30
PT	17	1	556	34	903	57

Table 1: Statistics of full and evaluation datasets.

4.1.2 Pre-Processing

In some studies, pre-processing is applied in order to improve the performance of CWS models, including substituting consecutive digits and English letters, Chinese idioms and long words with unique symbols (Cai and Zhao, 2016; Cai et al., 2017; Chen et al., 2015). However, we do not deploy such techniques for fair comparison, focusing only on the possible improvements brought by word embeddings. The only pre-processing adopted in our model is to first split a sentence with a set of pre-defined delimiters: characters that are not Chinese characters, English letters or digits. Those fragments of a sentence are then fed into the segmenter, and a complete segmented sentence is returned by reassembling the segmented fragments and delimiters in the original order.

4.1.3 Hyperparameters

Hyperparameters used in our WEB-CWS model are explained and their values are displayed in Table 2. All hyperparameters are tuned on a small excerpt of ZX, which consists of 300 sentences (Qiu and Zhang, 2015). It is worth noting that, according to Mikolov et al. (2013), for each positive sample, the optimal number of negative samples drawn from a noise distribution is usually between 5 to 20. However, in our model, we find that, for each target word, drawing one negative sample from a noise distribution is good enough, which may be caused by the large amount of negative samples generated by context and in-word negative sampling. Also, Mikolov et al. (2013) report that the unigram distribution raised to the 3/4ths power is better than the uniform distribution for negative sampling. But in WEB-CWS, using the uniform distribution leads to better segmentation results.

H-param	Explanation	Value
ϵ	threshold for overall subsampling	10^{-5}
μ	threshold for multi-character word subsampling	0.5
P_n	noise distribution for general negative sampling	uniform
n	number of general negative samples per word	1
d	dimension of word embeddings	100
η	smoothing factor for class weights	0.2
f	window size	4
m	initial maximum word length	5
k	initial beam-size	10
e	number of epochs in Skip-gram training	1

Table 2: Hyperparameters used in WEB-CWS.

4.1.4 Evaluation

For consistency, all segmentation results are automatically calculated with the script provided in the SIGHAN Bakeoff (Emerson, 2005) and are reported as word F-measures.

4.2 Baseline Segmenter

Two state-of-the-art CWS models trained on a People's Daily corpus in 2000 January are tested. One is a joint word segmentation and POS-tagging model (Zhang and Clark, 2010), and the other is a word-based neural CWS model (Cai et al., 2017). When training both models, default settings are used, except that the maximum word length in Cai et al.'s model is set to 5, which is in line with the setting of WEB-CWS.

On the evaluation set of PKU (Emerson, 2005), both models yield comparable results, but on the evaluation set of DL, Zhang and Clark's model (F-measure = 0.905) performs better than Cai et al.'s model (F-measure = 0.849). It is very possible that Zhang and Clark's model can handle cross-domain CWS more effectively. As a result, we choose Zhang and Clark's model as the baseline segmenter for following experiments.

5 Results

Results in Table 3 show that our WEB-CWS model can obviously improve CWS on four datasets in special domains, including DL, FR, ZX and DM, with an increase of over 3.0% in F-measure. Those four datasets are all in domains (novel and dermatology) which are very different from that of the baseline segmenter (newspaper). This result suggests that WEB-CWS can effectively improve cross-domain CWS.

However, on another dataset in the special domain PT, the improvement is not significant. There are two possible reasons for this result. First, the size of PT is the smallest among all datasets, which may make the quality of word embeddings unsatisfactory. Second, the PT dataset contains a huge amount of decimal points (e.g., '3.14'), percentage signs (e.g., '28%'), hyphens (e.g., 'pMIV-Pnlp') and very long English strings (e.g., 'agct-gagtcg'), which are all cases that cannot be handled by WEB-CWS without corresponding preprocessing techniques.

5.1 Comparison with State-of-the-Art Models

We also compare WEB-CWS with two state-of-the-art semi-supervised and unsupervised cross-domain CWS models by Qiu and Zhang (2015) and Liu and Zhang (2012), both of which use the same baseline model proposed by Zhang and Clark (2010) as used in our model. We adopt the method of combining character clustering and self-training in Liu and Zhang (2012) with datasets in our experiments. Results of the model in Qiu and Zhang (2015) are directly copied from the corresponding paper.

Results in Table 3 show that WEB-CWS outperforms these two state-of-the-art models with a large margin in terms of F-measure. Particularly, on the DM dataset, Liu and Zhang's model only achieves a relatively low F-score improvement rate (1.8%), which is likely to be caused by the large difference between the source and target domains. This result suggests that WEB-CWS is more robust to domain dissimilarity compared with self-training.

5.2 Run Time for Decoding

We test the run time for our decoder on a 3.5 GHz Intel Core i7 CPU. On all five test sets, the average decoding speed is 20.3 tokens per millisecond, when the initial beam-size is set to 10. In the work of Zhou et al. (2017), the decoding speed of their model is 14.7 tokens per millisecond for greedy segmentation. However, these results cannot be compared directly since they are produced on different machines. Our similarity look-up strategy is proved to be efficient in improving the decoding speed.

6 Analysis and Discussion

In order to assess the effect of negative sampling and subsampling methods, we conduct a series of

Dataset	Baseline (%)	WEB-CWS (%)	IR_WEB (%)	IR_Qiu&Zhang (%)	IR_Liu&Zhang (%)
DL	90.5	**93.5**	**+3.3**	+1.6	+2.1
FR	85.9	**89.6**	**+4.3**	+2.9	+2.5
ZX	86.8	**89.6**	**+3.2**	+2.2	+2.4
DM	77.9	**82.2**	**+5.5**	-	+1.8
PT	84.6	**85.1**	**+0.6**	-	+0.2

Table 3: F-measures of the baseline segmenter and WEB-CWS on datasets in special domains, and F-measure improvement rates (IR) of WEB-CWS, Qiu and Zhang (2015) and Liu and Zhang (2012).

Model	DL (%)	FR (%)	DM (%)
Baseline	90.5	85.9	77.9
WEB-basic	76.2	72.2	66.6
WEB + c_n	92.7	89.3	75.8
WEB + w_n	93.0	89.5	76.2
WEB + c_w_n	93.3	89.2	79.7
WEB + c_w_n + m_s	**93.5**	**89.6**	**82.2**

Table 4: F-measures of WEB-CWS with different subsampling and negative sampling methods.

Word	Meaning	Pre_base (%)	Pre_WEB (%)
点头	nod	0.0	**84.6**
兽武魂	**beast warrior spirit**	18.2	**100.0**
糖皮质激素	**glucocorticoid**	3.8	**76.9**
蛮胡子	**person name**	28.2	**97.4**
虚天鼎	**person name**	25.7	**94.3**
一对一	one on one	33.3	**100.0**
见于	to be seen in	34.5	**100.0**
鸡冠蛇	**cockscomb snake**	8.7	**73.9**
皮损	**skin lesions**	28.6	**92.9**
宁荣荣	**person name**	34.9	**98.8**

Table 5: Ten most improved words in terms of segmentation precision. Pre_base: segmentation precision by the baseline segmenter. Pre_WEB: segmentation precision by WEB-CWS.

ablation experiments. A detailed analysis is presented to understand in what way WEB-CWS can improve cross-domain CWS. All experiments and analyses in this section are carried out on three datasets with most significant improvements in F-measure: DL, FR and DM.

6.1 Ablation Experiments

In ablation experiments, we study the influence of two CWS-oriented negative sampling and multi-character words subsampling. Results in Table 4 show that WEB-CWS using word embeddings derived with the basic Skip-gram model ('*basic*') performs obviously worse than the baseline segmenter. When CWS-oriented negative sampling is applied alone, either context ('*c_n*') or in-word ('*w_n*') negative sampling, the performance of WEB-CWS is obviously better than or similar to that of the baseline segmenter. When both CWS-oriented negative sampling methods are applied together ('*c_w_n*'), WEB-CWS is ensured to obviously outperform the baseline segmenter. Also, when multi-character subsampling ('*m_s*') is applied, the performance of WEB-CWS can further improve a little.

6.2 Improvements in Noun Entity Segmentation

To see which words are incorrectly segmented by the baseline segmenter but correctly by WEB-CWS, all words occurring at least ten times in the three datasets are sorted in a descending order, by improvements in terms of segmentation precision. Table 5 displays the ten most improved words. As shown in Table 5, among the ten most improved words, seven words are domain-specific noun entities, including person names, disease names and chemical compound names. For some noun entities (e.g., *glucocorticoid*), even if the baseline segmenter can rarely segment them correctly, WEB-CWS can still find the correct segmentation in most cases. This result suggests that WEB-CWS is especially effective in segmenting domain-specific noun entities.

7 Conclusion

We have proposed WEB-CWS, a semi-supervised model that can be used to effectively improve cross-domain CWS. Our model only requires a baseline segmenter and a raw corpus in the target domain, deploying only word embeddings for CWS. WEB-CWS obviously improves the performance of the state-of-the-art baseline segmenter on four datasets in special domains, especially in segmenting domain-specific noun entities.

Acknowledgments

This paper was partially supported by the National Natural Science Foundation of China (No.

61572245). Thanks to You Wang et al. and the anonymous reviewers for their constructive and insightful comments on this paper.

References

Deng Cai and Hai Zhao. 2016. Neural word segmentation learning for chinese. *arXiv preprint arXiv:1606.04300*.

Deng Cai, Hai Zhao, Zhisong Zhang, Yuan Xin, Yongjian Wu, and Feiyue Huang. 2017. Fast and accurate neural word segmentation for chinese. *arXiv preprint arXiv:1704.07047*.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. 2015. Long short-term memory neural networks for chinese word segmentation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1197–1206.

Xinchi Chen, Zhan Shi, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-criteria learning for chinese word segmentation. *arXiv preprint arXiv:1704.07556*.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8. Association for Computational Linguistics.

Xiaolong Deng and Yingfei Sun. 2018. An improved embedding matching model for chinese word segmentation. In *2018 International Conference on Artificial Intelligence and Big Data (ICAIBD)*. IEEE.

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

G David Forney. 1973. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278.

Qin Gao and Vogel Stephan. 2010. A multi-layer chinese word segmentation system optimized for out-of-domain tasks. In *CIPS-SIGHAN Joint Conference on Chinese Language Processing*.

Yoav Goldberg and Omer Levy. 2014. Word2vec explained: Deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.

Yang Liu and Yue Zhang. 2012. Unsupervised domain adaptation for joint segmentation and pos-tagging. *Proceedings of COLING 2012: Posters*, pages 745–754.

Yijia Liu, Yue Zhang, Wanxiang Che, Ting Liu, and Fan Wu. 2014. Domain adaptation for crf-based chinese word segmentation using free annotations. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 864–874.

Jianqiang Ma and Erhard Hinrichs. 2015. Accurate linear-time chinese word segmentation via embedding matching. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1733–1743.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive science*, 34(8):1388–1429.

Mark EJ Newman. 2005. Power laws, pareto distributions and zipf's law. *Contemporary physics*, 46(5):323–351.

Fuchun Peng, Fangfang Feng, and Andrew McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. In *Proceedings of the 20th international conference on Computational Linguistics*, page 562. Association for Computational Linguistics.

Likun Qiu, Linlin Shi, and Houfeng Wang. 2015. Construction of chinese dependency tree-bank and analysis of factors in statistical parsing in various domains. *Journal of Chinese Information Processing*, 29(5):69–76.

Likun Qiu and Yue Zhang. 2015. Word segmentation for chinese novels. In *AAAI*, pages 2440–2446.

Sunita Sarawagi and William W Cohen. 2005. Semi-markov conditional random fields for information extraction. In *Advances in neural information processing systems*, pages 1185–1192.

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

Guohua Wu, Dezhu He, Keli Zhong, Xue Zhou, and Caixia Yuan. 2014. Leveraging rich linguistic features for cross-domain chinese segmentation. In *Proceedings of The Third CIPS-SIGHAN Joint Conference on Chinese Language Processing*, pages 101–107.

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

Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pages 840–847.

Yue Zhang and Stephen Clark. 2010. A fast decoder for joint word segmentation and pos-tagging using a single discriminative model. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 843–852. Association for Computational Linguistics.

Hai Zhao, Deng Cai, Yang Xin, Yuzhu Wang, and Zhongye Jia. 2017. A hybrid model for chinese spelling check. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, 16(3):21.

Hao Zhou, Zhenting Yu, Yue Zhang, Shujian Huang, XIN-YU DAI, and Jiajun Chen. 2017. Word-context character embeddings for chinese word segmentation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 760–766.