Regularized Structured Perceptron: A Case Study on Chinese Word Segmentation, POS Tagging and Parsing

Kaixu Zhang
Xiamen University
Fujian, P.R. China
kareyzhang@gmail.com

Jinsong Su
Xiamen University
Fujian, P.R. China
jssu@xmu.edu.cn

Changle Zhou
Xiamen University
Fujian, P.R. China
dozero@xmu.edu.cn

Abstract

Structured perceptron becomes popular for various NLP tasks such as tagging and parsing. Practical studies on NLP did not pay much attention to its regularization. In this paper, we study three simple but effective task-independent regularization methods: (1) one is to average weights of different trained models to reduce the bias caused by the specific order of the training examples; (2) one is to add penalty term to the loss function; (3) and one is to randomly corrupt the data flow during training which is called dropout in the neural network. Experiments are conducted on three NLP tasks, namely Chinese word segmentation, part-of-speech tagging and dependency parsing. Applying proper regularization methods or their combinations, the error reductions with respect to the averaged perceptron for some of these tasks can be up to 10%.

1 Introduction

Structured perceptron is a linear classification algorithm. It is used for word segmentation (Zhang and Clark, 2011), POS (part-of-speech) tagging (Collins, 2002), syntactical parsing (Collins and Roark, 2004), semantical parsing (Zettlemoyer and Collins, 2009) and other NLP tasks.

The averaged perceptron or the voted perceptron (Collins, 2002) is proposed for better generalization. Early update (Collins and Roark, 2004; Huang et al., 2012) is used for inexact decoding algorithms such as the beam search. Distributed training (McDonald et al., 2010) and the minibatch and parallelization method (Zhao and Huang, 2013) are recently proposed. Some other related work focuses on the task-specified feature engineering.

Regularization is to improve the ability of generalization and avoid over-fitting for machine learning algorithms including online learning algorithms (Do et al., 2009; Xiao, 2010). But practical studies on NLP did not pay much attention to the regularization of the structured perceptron. As a result, for some tasks the model learned using perceptron algorithm is not as good as the model learned using regularized condition random field.

In this paper, we treat the perceptron algorithm as a special case of the stochastic gradient descent (SGD) algorithm and study three kinds of simple but effective task-independent regularization methods that can be applied. The *averaging* method is to average the weight vectors of different models. We propose a "shuffle-and-average" method to reduce the bias caused by the specific order of the training examples. The traditional *penalty* method is to add penalty term to the loss function. The *dropout* method is to randomly corrupt the data flow during training. We show that this dropout method originally used in neural network also helps the structured perceptron.

In Section 2, we describe the perceptron algorithm as a special case of the stochastic gradient descent algorithm. Then we discuss three kinds of regularization methods for structured perceptron in Section 3, 4 and 5, respectively. Experiments conducted in Section 6 shows that these regularization methods and their combinations improve performances of NLP tasks such as Chinese word segmentation, POS tagging and dependency parsing. Applying proper regularization methods, the error reductions of these NLP tasks can be up to 10%. We finally conclude this work in Section 7.

Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, pages 164–173,
Gothenburg, Sweden, April 26-30 2014.

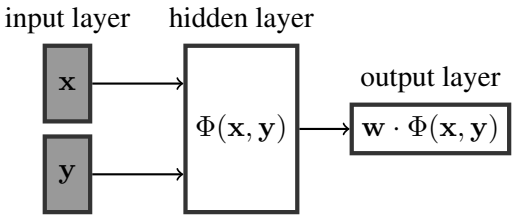


Figure 1: A structured perceptron can be seen as a multi-layer feed-forward neural network.

2 Structured Perceptron

We treat the structured perceptron architecture as a multi-layer feed-forward neural network as in Figure 1 and treat the perceptron algorithm as a special case of the stochastic gradient descent algorithm in order to describe all the regularization methods.

The network of the structured perceptron has three layers. The input vector $\mathbf{x}$ and output vector $\mathbf{y}$ of the structured classification task are concatenated as the input layer. The hidden layer is the feature vector $\Phi(\mathbf{x}, \mathbf{y})$. The connections between the input layer and the hidden layer are usually hand-crafted and fixed during training and predicting. And the output layer of the network is a scalar $\mathbf{w} \cdot \Phi(\mathbf{x}, \mathbf{y})$ which is used to evaluate the matching of the vector $\mathbf{x}$ and $\mathbf{y}$.

Besides the common process to calculate the output layer given the input layer, there is a process called decoding, which is to find a vector $\mathbf{z}$ to maximum the activation of the output layer:

$$\mathbf{z}_i = \arg\max_{\mathbf{z}} \mathbf{w} \cdot \Phi(\mathbf{x}_i, \mathbf{z}) \tag{1}$$

By carefully designing the feature vector, the decoding can be efficiently performed using dynamic programming. Beam search is also commonly used for the decoding of syntactical parsing tasks.

In the predicting precess, the vector $\mathbf{z}$ is the structured output corresponding to $\mathbf{x}$. In the training precess, what we expect is that for every input $\mathbf{x}_i$, the vector $\mathbf{z}_i$ that maximums the activation of the output layer is exactly the gold standard output $\mathbf{y}_i$.

We define the loss function as the sum of the margins of the whole training data:

$$\begin{aligned}\mathcal{L}(\mathbf{w}) &= \sum_i \{\mathbf{w} \cdot \Phi(\mathbf{x}_i, \mathbf{z}_i) - \mathbf{w} \cdot \Phi(\mathbf{x}_i, \mathbf{y}_i)\} \\ &= \mathbf{w} \sum_i \cdot \Delta\Phi_i\end{aligned} \tag{2}$$

where

$$\Delta\Phi_i = \Phi(\mathbf{x}_i, \mathbf{z}_i) - \Phi(\mathbf{x}_i, \mathbf{y}_i) \tag{3}$$

The unconstrained optimization problem of the training process is

$$\arg\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}) \tag{4}$$

The loss function is not convex but calculating the derivative is easy. One of the algorithms to solve this optimization problem is SGD. Here we use the minibatch with size of 1, which means in every iteration we use only one training example to approximate the loss function and the gradient to update the weight vector:

$$\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \eta \left.\frac{\partial \mathcal{L}}{\partial \mathbf{w}}\right|_{\mathbf{w}^{(t)}} \approx \mathbf{w}^{(t)} - \eta\Delta\Phi^{(t)} \tag{5}$$

where $\mathbf{w}^{(t)}$ is the weight vector after t updates. Note that in this case, the learning rate η can be set to an arbitrary positive real number. In the perceptron algorithm commonly used in NLP (Collins, 2002) , η is not changed respect to t. We fix η to be 1 in this paper without loss of generality.

3 Averaging

3.1 Averaged Perceptron

Averaging the weight vectors in the learning process is one of the most popular regularization techniques of the structured perceptron (Collins, 2002). And it is also the only used regularization technique for many practical studies on NLP (Jiang et al., 2009; Huang and Sagae, 2010).

Suppose the learning algorithm stopped after T updates. The final weight vector is calculated as:

$$\mathbf{w} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{w}^{(t)} \tag{6}$$

The intuition might be that the learned weight vector is dependent on the order of the training examples. The final vector $\mathbf{w}^{(T)}$ may be more appropriate for the last few training examples than the previous ones. The averaging method is used to avoid such tendency. Similar treatment is used in other sequential algorithm such as the Markov chain Monte Carlo sampling method.

Since this regularization technique is widely used and tested, it is used for all the models in the experiments of this paper. Any other regularization methods are applied to this basic averaged perceptron.

3.2 Shuffle and Average

As we has mentioned that the learned weight vector is strongly dependent on the order of the training examples, randomly shuffling the training examples results in different weight vectors. Based on such observation, we training different weight vectors using the same training examples with different orders, and average them to get the final weight vector. We use this method to further minimize the side effect caused by this online algorithm.

Suppose we shuffle and train n different weight vectors $\mathbf{w}^{[1]}, \ldots, \mathbf{w}^{[n]}$, the j-th component of the final vector can be simply calculated as

$$w_j = \frac{\sum_{i=1}^{n} w_j^{[i]}}{n} \tag{7}$$

Note that generally these models do not share the same feature set. Features may be used in one model but not in another one. When $w_j^{[i]} = 0$, it does not imply that this feature has no effect on this problem. It only implies that this feature does not have chances to be tested. We propose a modified equation to only average the non-zero components:

$$w_j = \frac{\sum_{i=1}^{n} w_j^{[i]}}{\left|\{i|w_j^{[i]} \neq 0, i = 1, \cdots, n\}\right|} \tag{8}$$

This equation makes the low-frequency features more important in the final model.

4 Penalty

Adding penalty term to the loss function is a common and traditional regularization method to avoid over-fitting. It is widely used for the optimization problems of logistic regression, support vector machine, conditional random field and other models. Penalty terms for probabilistic models can be interpreted as a prior over the weights (Chen and Rosenfeld, 1999). It is also called "weight decay" in artificial neural network (Moody et al., 1995). The use of the penalty term is to prevent the components of the weight vector to become too large.

In Section 2 we have modeled the perceptron algorithm as an SGD algorithm with an explicit loss function, the additional penalty term is therefore easy to be employed.

4.1 L2-norm penalty

We can add a square of the L2-norm of the weight vector as the penalty term to the loss function as

$$\mathcal{L} = \mathbf{w} \cdot \sum_i \Delta\Phi_i + \frac{\lambda_2}{2}\|\mathbf{w}\|_2^2 \tag{9}$$

where λ_2 is a hyper-parameter to determine the strength of the penalty.

In the SGD algorithm, the update method of the weight vector is thus

$$\mathbf{w}^{(t+1)} \leftarrow (1 - \eta\lambda_2)\mathbf{w}^{(t)} - \eta\Delta\Phi^{(t)} \tag{10}$$

The term $(1 - \eta\lambda_2)$ is used to decay the weight in every updates. This forces the weights to be close to zero.

4.2 L1-norm penalty

Another commonly used penalty term is the L1-norm of the weight vector. This kinds of terms usually results in sparse weight vector. Since the averaged perceptron is used, the final averaged weight vector will not be sparse.

The loss function using the L1-nrom penalty is

$$\mathcal{L} = \mathbf{w} \cdot \sum_i \Delta\Phi_i + \lambda_1\|\mathbf{w}\|_1 \tag{11}$$

where λ_1 is the hyper-parameter to determine the strength of the penalty.

The derivative of the penalty term is discontinuous. We update the weights as

$$w_i^{(t+1)} \leftarrow \frac{\max\{0, |w_i^{(t)}| - \eta\lambda_1\}}{|w_i^{(t)}|} w_i^{(t)} - \eta\Delta\phi_i^{(t)} \tag{12}$$

This ensures that the weight decay will not change the sign of the weight.

An modified version of the L1 penalty for the online learning is the cumulative L1 penalty (Tsuruoka et al., 2009), which is used to make the stochastic gradient of the penalty term more close to the true gradient. The update is divided into two steps. In the first step, the weight vector is updated according to the loss function without the penalty term

$$w_i^{(t+\frac{1}{2})} \leftarrow w_i^{(t)} - \eta\Delta\phi_i^{(t)} \tag{13}$$

And the cumulative penalty is calculated separately

$$c_i^{(t+\frac{1}{2})} \leftarrow c_i^{(t)} + \eta\lambda_1 \tag{14}$$

In the second step, $|w_i|$ and c_i are compared and at most one of them is non-zero before the next update

$$m \leftarrow \min\{|w_i^{(t+\frac{1}{2})}|, c_i^{(t+\frac{1}{2})}\} \quad (15)$$

$$w_i^{(t+1)} \leftarrow \frac{\max\{0, |w_i^{(t+\frac{1}{2})}| - m\}}{|w_i^{(t+\frac{1}{2})}|} w_i^{(t+\frac{1}{2})} \quad (16)$$

$$c_i^{(t+1)} \leftarrow c_i^{(t+\frac{1}{2})} - m \quad (17)$$

5 Dropout

Dropout (Hinton et al., 2012) is originally a regularization method used for the artificial neural network. It corrupts one or more layers of a feed-forward network during training, by randomly omitting some of the neurons. If the input layer is corrupted during the training of an autoencoder, the model is called denoising autoencoder (Vincent et al., 2008).

The reason why such treatment can regularize the parameters are explained in different ways. Hinton et al. (2012) argued that the final model is an average of a large number of models and the dropout forces the model to learn good features which are less co-adapted. Vincent et al. (2008) argued that by using dropout of the input layer, the model can learn how to deal with examples outside the low-dimensional manifold that the training data concentrate.

Models not so deep such as the structured perceptron may also benefit from this idea. Following the dropout method used in neural network, we give the similar method for structured perceptron.

5.1 Input Layer

We can perform dropout for structured perceptron by corrupting the input layer in Figure 1. Since we concern that what $\mathbf{y}$ exactly is, we only corrupt $\mathbf{x}$. The components of the corrupted vector $\tilde{\mathbf{x}}$ is calculated as

$$\tilde{x}_i = x_i n_i \quad (18)$$

where $n_i \sim \text{Bern}(p)$ obey a Binomial distribution with the hyper-parameter p.

During training, the decoding processing with the corrupted input is

$$\mathbf{z} = \arg\max_{\mathbf{z}} \mathbf{w} \cdot \Phi(\tilde{\mathbf{x}}, \mathbf{z}) \quad (19)$$

The $\mathbf{x}$ in the loss function is also substituted with the corrupted version $\tilde{\mathbf{x}}$.

Note that the corruption decreases the number of non-zero components of the feature vector Φ, which makes the decoding algorithm harder to find the gold standard $\mathbf{y}$.

For NLP tasks, the input vector $\mathbf{x}$ could be a sequence of tokens (words, POS tags, etc.). The corruption substitutes some of the tokens with a special token null. Any features contain such token will be omitted (This is also the case for the out-of-vocabulary words during predicting). So the dropout of $\mathbf{x}$ in NLP during training can be explained as to randomly mask some of the input tokens. The decoder algorithm needs to find out the correct answer even if some parts of the input are unseen. This harder situation could force the learning algorithm to learn better models.

5.2 Hidden Layer

The dropout can also be performed at the hidden layer. Likewise, the components of the corrupted feature vector $\tilde{\Phi}$ is calculated as

$$\tilde{\phi}_i = \phi_i m_i \quad (20)$$

where $m_i \sim \text{Bern}(q)$ obey a Binomial distribution with the hyper-parameter q.

The Φ in the decoding processing during training and the loss function is substituted with $\tilde{\Phi}$.

6 Experiments

In this section, we first introduce three NLP tasks using structured perceptron namely Chinese word segmentation, POS tagging and dependency parsing. Then we investigate the effects of regularization methods for structured perceptron mainly on the development set of character-based Chinese word segmentation. Finally, we compare the final performances on the test sets of these three tasks using regularization methods with related work.

6.1 Tasks

6.1.1 Chinese Word Segmentation

A Chinese word consists of one or more Chinese characters. But there is no spaces in the sentences to indicating words. Chinese word segmentation is the task to segment words in the sentence.

We use a character-based Chinese word segmentation model as the baseline. Like part-of-speech tagging which is to assign POS tags to words sequence, character-based Chinese word segmentation is to assign tags to character sequence. The tag set of four tags is commonly used:

Type	Templates
Unigram	$\langle x_{i-1}, y_i\rangle$, $\langle x_i, y_i\rangle$, $\langle x_{i+1}, y_i\rangle$
Bigram	$\langle x_{i-2}, x_{i-1}, y_i\rangle$, $\langle x_{i-1}, x_i, y_i\rangle$
	$\langle x_i, x_{i+1}, y_i\rangle$, $\langle x_{i+1}, x_{i+2}, y_i\rangle$
transition	$\langle y_{i-1}, y_i\rangle$

Table 1: Feature templates for the character-based Chinese word segmentation model and the joint Chinese word segmentation and POS tagging model.

tag S indicates that the character forms a single-character words; tag B / E indicates that the character is at the beginning / end of a multi-character words; tag M indicates that the character is in the middle of a multi-character words.

For example, if the tag sequence for the input

$$\mathbf{x} = \text{菊次郎的夏天} \tag{21}$$

is

$$\mathbf{y} = \text{BMESBE}, \tag{22}$$

the corresponding segmentation result is

$$\text{菊次郎 的 夏天}. \tag{23}$$

Table 1 shows the set of the feature templates which is a subset of some related work (Ng and Low, 2004; Jiang et al., 2009) .

Following Sun (2011), we split the Chinese treebank 5 into training set, development set and test set. F-measure (Emerson, 2005) is used as the measurement of the performance.

6.1.2 Part-of-Speech Tagging

The second task is joint Chinese word segmentation and POS tagging. This can also be modeled as a character-based sequence labeling task.

The tag set is a Cartesian product of the tag set for Chinese word segmentation and the set of POS tags. For example, the tag B-NN indicates the character is the first character of a multi-character noun. The tag sequence

$$\mathbf{y} = \text{B-NR M-NR E-NR S-DEG B-NN E-NN}, \tag{24}$$

for the input sentence in Equation (21) results in

$$\text{菊次郎\_NR 的\_DEG 夏天\_NN}. \tag{25}$$

The same feature templates shown in Table 1 are used for joint Chinese word segmentation and POS tagging.

Also, we use the same training set, development set and test set based on CTB5 corpus as the Chinese word segmentation task. F-measure for joint Chinese word segmentation and POS tagging is used as the measurement of the performance.

6.1.3 Dependency Parsing

The syntactical parsing tasks are different with previously introduced tagging tasks. To investigate the effects of regularization methods on the parsing tasks, we fully re-implement the linear-time incremental shift-reduce dependency parser by Huang and Sagae (2010). The structure perceptron is used to train such model. The model totally employs 28 feature templates proposed by Huang and Sagae (2010).

Since the search space for parsing tasks is quite larger than the search space for tagging tasks, Exact search algorithms such as dynamic programming can not be used. Besides, beam search with state merging is used for decoding. The early update strategy (Collins and Roark, 2004) is also employed.

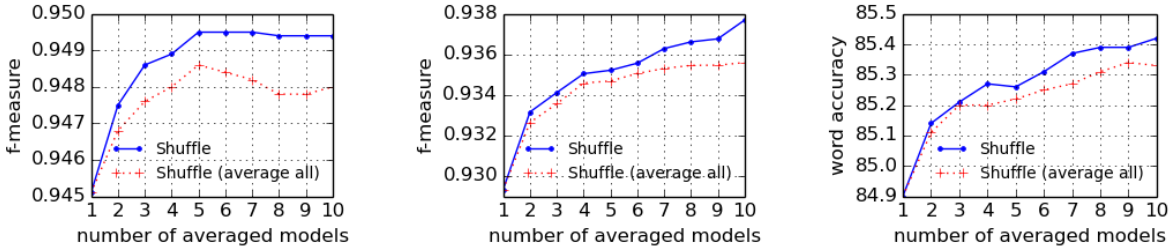
In order to compare to the related work, unlike the Chinese word segmentation and the POS tagging task, we split the CTB5 corpus following Zhang et al.(2008). Two types of accuracies are used to measure the performances, namely word and complete match (excluding punctuations) (Huang and Sagae, 2010).

6.2 Averaging

First, we investigate the effect of averaging techniques for regularization. Figure 2 shows the influence of the number of the averaged models by using the "shuffle-and-average" method described in section 3.2. The performances of the Chinese word segmentation, POS tagging and parsing tasks are all increased by averaging models trained with the same training data with different orders. The "shuffle-and-average" method is effective to reduce the bias caused by the specific order of the training examples.

For the Chinese word segmentation task which is a relatively simple task, averaging about five different models can achieve the best effect; whereas for POS tagging and parsing, averaging more models will continually increase the performance even when the number of models approaches 10.

The dotted lines in Figure 2 indicate the performances by using Equation (7) for model averaging. The solid lines indicate the performances by



(a) Chinese word segmentation (b) POS tagging (c) Dependency parsing

Figure 2: The influence of the number of the averaged models using the "shuffle-and-average" method for (a) Chinese word segmentation, (b) POS tagging and (c) dependency parsing. "Shuffle" means to only average the non-zero weights (Equation (8)), while "Shuffle (average all)" means to average all weights (Equation (7)).

using Equation (8) for model averaging. According to these three different tasks, Equation (8) always performs better than Equation (7). We will use Equation (8) denoted as "Shuffle" for the rest of the experiments.

6.3 Penalty

Here we investigate the penalty techniques for regularization only using the character-based Chinese word segmentation task.

Figure 3 shows the effect of adding L1-norm and L2-norm penalty terms to the loss function.

With appropriate hyper-parameters, the performances are increased. According to the performances, adding L2 penalty is slightly better than adding L1 penalty or adding cumulative L1 penalty.

We then combine the "shuffle-and-average" method with the penalty methods. The performances (solid lines in Figure 3) are further improved and are better than those of models that only use one regularization method.

6.4 Dropout

We also investigate the dropout method for regularization using the character-based Chinese word segmentation task.

Figure 4 shows the effect of the dropout method ("dropout" for the input layer and "dropout (Φ)" for the hidden layer) and the combination of the dropout and "shuffle-and-average" method (solid line). We observed that the dropout for the hidden layer is not effective for structured perceptron. This may caused by that the connections between the input layer and the hidden layer are fixed during training. Neurons in the hidden layer can not

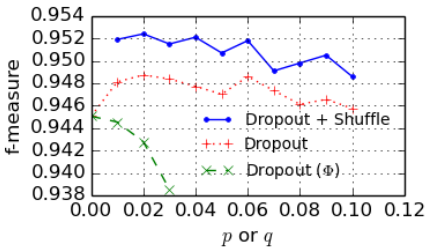


Figure 4: Influences of the hyper-parameter p (for the input layer, denoted as "dropout") or q (for the hidden layer, denoted as "dropout (Φ)") for the dropout method.

changes the weights to learn different representations for the input layer. On the other hand, the dropout for the input layer improves the performance. Combining the dropout and the "shuffle-and-average" method, the performance is further improved.

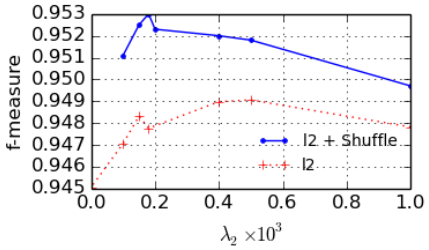
Figure 5 shows the effect of the combination of the three regularization methods. We see that no matter what other regularization methods are already used, adding "shuffle-and-average" method can always improve the performance. The effects of the penalty method and the dropout method have some overlap, since combining these two method does not result in a significant improvement of the performance.

6.5 Final Results

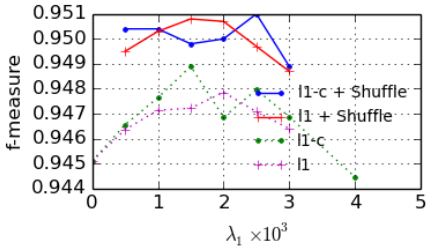
6.5.1 Chinese Word Segmentation

Table 2 shows the final results of the character-based Chinese word segmentation task on the test set of the CTB5 corpus.

Structure perceptron with feature templates in



(a) L2-norm penalty



(b) L1-norm penalty

Figure 3: influence of the hyper-parameter λ_2 in the L2-norm penalty term and λ_1 in the L1-norm penalty term ("l1-c" indicates the cumulative L1 penalty) for the character-based Chinese word segmentation task.

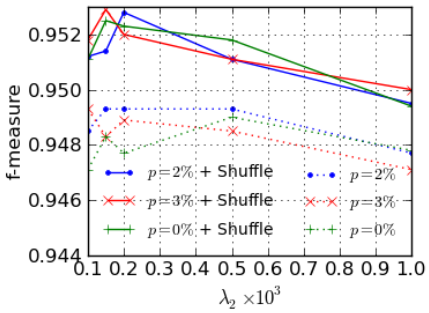


Figure 5: The combination of these three regularization methods.

Table 1 is used. We use the "shuffle-and-average" (5 models), the L2 penalty method ($\lambda_2 = 10^{-4}$), the dropout method ($p = 3\%$) and their combinations to regularize the structured perceptron.

To compare with the perceptron algorithm, we use the conditional random field model (CRF) with the same feature templates in Table 1 to train the model parameters. The toolkit CRF++[1] with the L2-norm penalty is used to train the weights. The hyper-parameter $C = 20$ is tuned using the development set.

Jiang et al. (2009) proposed a character-based model employing similar feature templates using averaged perceptron. The feature templates are following Ng and Low (2004). Zhang and Clark (2011) proposed a word-based model employing both character-based features and more sophisticated word-based features using also averaged perceptron. There are other related results (Jiang et al., 2012) of open test including the final result of Jiang et al. (2009). Since their models used extra resources, they are not comparable with the results in this paper.

[1] http://crfpp.googlecode.com/svn/trunk/doc/index.html

	sf
(Jiang et al., 2009)	0.9735
(Zhang and Clark, 2011)	0.9750†
CRF++ ($C = 20$)	0.9742
Averaged Percetron	0.9734
+ Shuffle	0.9755
+ L2	0.9736
+ L2 + Shuffle	**0.9772**
+ Dropout	0.9741
+ Dropout+ Shuffle	0.9765
+ L2 + Dropout	0.9749
+ L2 + Dropout+ Shuffle	0.9771

Table 2: Final results of the character-based Chinese word segmentation task on CTB5. † This result is read from a figure in that paper.

	sf
Word-based model	0.9758
+ Shuffle	0.9787
+ L2 + Shuffle	**0.9791**
+ L2 + Dropout+ Shuffle	**0.9791**

Table 3: Final results of the word-based Chinese word segmentation task on CTB5.

The results in Table 2 shows that with proper regularization methods, the models trained using perceptron algorithm can outperform CRF models with the same feature templates and other models with more sophisticated features trained using the averaged perceptron without other regularization methods.

We further re-implemented a word-based Chinese word segmentation model with the feature templates following Zhang et al. (2012), which

	sf	jf
(Jiang et al., 2008)	0.9785	0.9341
(Kruengkrai et al., 2009)	0.9787	0.9367
(Zhang and Clark, 2010)	0.9778	0.9367
(Sun, 2011)	0.9817	0.9402
Character-based model	0.9779	0.9336
+ Shuffle	0.9802	0.9375
+ Dropout	0.9789	0.9361
+ Dropout+ Shuffle	0.9809	0.9407
+ word-based re-ranking	0.9813	**0.9438**

Table 4: Final results of the POS tagging task on CTB5.

	word	compl.
(Huang and Sagae, 2010)	85.20	33.72
our re-implementation	85.22	34.15
+ Shuffle	85.65	34.52
+ Dropout	85.32	34.04
+ Dropout+ Shuffle	**85.71**	34.57

Table 5: Final results of the dependency parsing task on CTB5.

is similar with the model proposed by Zhang and Clark (2011). Beam search with early-update is used for decoding instead of dynamic programming. The results with different regularization methods are shown in Figure 3. These regularization methods show similar characteristics for the word-based model.

6.5.2 POS Tagging

The results of the POS tagging models on the CTB5 corpus are shown in Table 4. Structure perceptron with feature templates in Table 1 is used. The F-measures for word segmentation (sf) and for joint word segmentation and POS tagging (jf) are listed.

We use the "shuffle-and-average" (10 models), the dropout method ($p = 5\%$) and their combination to regularize the structured perceptron.

Jiang et al. (2008) used a character-based model using perceptron for POS tagging and a log-linear model for re-ranking. Kruengkrai et al. (2009) proposed a hybrid model including character-based and word-based features. Zhang and Clark (2010) proposed a word-based model using perceptron. Sun (2011) proposed a framework based on stacked learning consisting of four sub-models. For the closed test, this model has the best performance on the CTB5 corpus to our knowledge. Other results (Wang et al., 2011; Sun and Wan, 2012) for the open test are not listed since they are not comparable with the results in this paper.

If we define the error rate as $1 - \text{jf}$, the error reduction by applying regularization methods for the character-based model is more than 10%. Comparing to the related work, the character-based model that we used is quite simple. But using the regularization methods discussed in this paper, it provides a comparable performance to the best model in the literature.

6.5.3 Dependency Parsing

Table 5 shows the final results of the dependency parsing task on the CTB5 corpus. We use the "shuffle-and-average" (10 models), the dropout method ($p = 5\%$ only for the words in the input) and their combination to regularize the structured perceptron based on Huang and Sagae's (2010).

The performance of the parsing model is also improved by using more regularization methods, although the improvement is not as remarkable as those for tagging tasks. For the parsing tasks, there are many other factors that impact the performance.

7 Conclusion

The "shuffle-and-average" method can effectively reduce the bias caused by the specific order of the training examples. It can improve the performance even if some other regularization methods are applied.

When we treat the perceptron algorithm as a special case of the SGD algorithm, the traditional penalty methods can be applied. And our observation is that L2 penalty is better than L1 penalty.

The dropout method is derived from the neural network. Corrupting the input during training improves the ability of generalization. The effects of the penalty method and the dropout method have some overlap.

Experiments showed that these regularization methods help different NLP tasks such as Chinese word segmentation, POS tagging and dependency parsing. Applying proper regularization methods, the error reductions for some of these NLP tasks can be up to 10%. We believe that these methods can also help other models which are based on structured perceptron.

Acknowledgments

The authors want to thanks all the reviews for many pertinent comments which have improved the quality of this paper. The authors are supported by NSFC (No. 61273338 and No. 61303082), the Doctoral Program of Higher Education of China (No. 20120121120046) and China Postdoctoral Science Foundation (No. 2013M541861).

References

Stanley F Chen and Ronald Rosenfeld. 1999. A gaussian prior for smoothing maximum entropy models. Technical report, DTIC Document.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, page 111118, Barcelona, Spain, July.

Michael Collins. 2002. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. pages 1–8.

Chuong B Do, Quoc V Le, and Chuan-Sheng Foo. 2009. Proximal regularization for online and batch learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 257–264. ACM.

Thomas Emerson. 2005. The second international chinese word segmentation bakeoff. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, pages 123–133. Jeju Island, Korea.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Liang Huang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1077–1086, Uppsala, Sweden, July. Association for Computational Linguistics.

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured perceptron with inexact search. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–151. Association for Computational Linguistics.

Wenbin Jiang, Liang Huang, Qun Liu, and Yajuan Lü. 2008. A cascaded linear model for joint chinese word segmentation and part-of-speech tagging. In *Proceedings of ACL-08: HLT*, pages 897–904, Columbus, Ohio, June. Association for Computational Linguistics.

Wenbin Jiang, Liang Huang, and Qun Liu. 2009. Automatic adaptation of annotation standards: Chinese word segmentation and POS tagging - a case study. In *Proceedings of the 47th ACL*, pages 522–530, Suntec, Singapore, August. Association for Computational Linguistics.

Wenbin Jiang, Fandong Meng, Qun Liu, and Yajuan Lü. 2012. Iterative annotation transformation with predict-self reestimation for chinese word segmentation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 412–420, Jeju Island, Korea, July. Association for Computational Linguistics.

Canasai Kruengkrai, Kiyotaka Uchimoto, Jun'ichi Kazama, Yiou Wang, Kentaro Torisawa, and Hitoshi Isahara. 2009. An error-driven word-character hybrid model for joint chinese word segmentation and POS tagging. In *Proc. of ACL-IJCNLP 2009*, pages 513–521, Suntec, Singapore. Association for Computational Linguistics.

Ryan McDonald, Keith Hall, and Gideon Mann. 2010. Distributed training strategies for the structured perceptron. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 456–464. Association for Computational Linguistics.

JE Moody, SJ Hanson, Anders Krogh, and John A Hertz. 1995. A simple weight decay can improve generalization. *Advances in neural information processing systems*, 4:950–957.

Hwee Tou Ng and Jin Kiat Low. 2004. Chinese part-of-speech tagging: One-at-a-time or all-at-once? word-based or character-based? In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 277–284, Barcelona, Spain, July. Association for Computational Linguistics.

Weiwei Sun and Xiaojun Wan. 2012. Reducing approximation and estimation errors for chinese lexical processing with heterogeneous annotations. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 232–241, Jeju Island, Korea, July. Association for Computational Linguistics.

Weiwei Sun. 2011. A stacked sub-word model for joint chinese word segmentation and part-of-speech tagging. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1385–1394, Portland, Oregon, USA, June. Association for Computational Linguistics.

Yoshimasa Tsuruoka, Jun'ichi Tsujii, and Sophia Ananiadou. 2009. Stochastic gradient descent training for l1-regularized log-linear models with cumulative penalty. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the*

4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1, pages 477–485. Association for Computational Linguistics.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, page 10961103.

Yiou Wang, Jun'ichi Kazama, Yoshimasa Tsuruoka, Wenliang Chen, Yujie Zhang, and Kentaro Torisawa. 2011. Improving chinese word segmentation and POS tagging with semi-supervised methods using large auto-analyzed data. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 309–317, Chiang Mai, Thailand, November. Asian Federation of Natural Language Processing.

Lin Xiao. 2010. Dual averaging methods for regularized stochastic learning and online optimization. *The Journal of Machine Learning Research*, 9999:2543–2596.

Luke S Zettlemoyer and Michael Collins. 2009. Learning context-dependent mappings from sentences to logical form. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 976–984. Association for Computational Linguistics.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, page 562571, Honolulu, Hawaii, October. Association for Computational Linguistics.

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, Cambridge, MA, October. Association for Computational Linguistics.

Y. Zhang and S. Clark. 2011. Syntactic processing using the generalized perceptron and beam search. *Computational Linguistics*, (Early Access):1–47.

Kaixu Zhang, Maosong Sun, and Changle Zhou. 2012. Word segmentation on chinese mirco-blog data with a linear-time incremental model. In *Proceedings of the Second CIPS-SIGHAN Joint Conference on Chinese Language Processing*, pages 41–46, Tianjin, China, December. Association for Computational Linguistics.

Kai Zhao and Liang Huang. 2013. Minibatch and parallelization for online large margin structured learning. In *Proceedings of NAACL-HLT*, pages 370–379.