Iterative Annotation Transformation with Predict-Self Reestimation for Chinese Word Segmentation

Wenbin Jiang and **Fandong Meng** and **Qun Liu** and **Yajuan Lü**

Key Laboratory of Intelligent Information Processing
Institute of Computing Technology
Chinese Academy of Sciences
{jiangwenbin, mengfandong, liuqun, lvyajuan}@ict.ac.cn

Abstract

In this paper we first describe the technology of automatic annotation transformation, which is based on the annotation adaptation algorithm (Jiang et al., 2009). It can automatically transform a human-annotated corpus from one annotation guideline to another. We then propose two optimization strategies, iterative training and predict-self reestimation, to further improve the accuracy of annotation guideline transformation. Experiments on Chinese word segmentation show that, the iterative training strategy together with predict-self reestimation brings significant improvement over the simple annotation transformation baseline, and leads to classifiers with significantly higher accuracy and several times faster processing than annotation adaptation does. On the Penn Chinese Treebank 5.0, it achieves an F-measure of 98.43%, significantly outperforms previous works although using a single classifier with only local features.

1 Introduction

Annotation guideline adaptation depicts a general pipeline to integrate the knowledge of corpora with different underling annotation guidelines (Jiang et al., 2009). In annotation adaptation two classifiers are cascaded together, where the classification results of the lower classifier are used as guiding features of the upper classifier, in order to achieve more accurate classification. This method can automatically adapt the divergence between different annotation guidelines and bring improvement to Chinese word segmentation. However, the need of cascaded classification decisions makes it less practical for tasks of high computational complexity such as parsing, and less efficient to incorporate more than two annotated corpora.

In this paper, we first describe the algorithm of automatic annotation transformation. It is based on the annotation adaptation algorithm, and it focuses on the automatic transformation (rather than adaptation) of a human-annotated corpus from one annotation guideline to another. First, a classifier is trained on the corpus with an annotation guideline not desired, it is used to classify the corpus with the annotation guideline we want, so as to obtain a corpus with parallel annotation guidelines. Then a second classifier is trained on the parallelly annotated corpus to learn the statistical regularity of annotation transformation, and it is used to process the previous corpus to transform its annotation guideline to that of the target corpus. Instead of the online knowledge integration methodology of annotation adaptation, annotation transformation can lead to improved classification accuracy in an offline manner by using the transformed corpora as additional training data for the classifier. This method leads to an enhanced classifier with much faster processing than the cascaded classifiers in annotation adaptation.

We then propose two optimization strategies, iterative training and predict-self reestimation, to further improve the accuracy of annotation transformation. Although the transformation classifiers can only be trained on corpora with autogenerated (rather than gold) parallel annotations, an iterative training procedure can gradually improve the trans-

Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 412–420, Jeju Island, Korea, 12–14 July 2012. ©2012 Association for Computational Linguistics

formation accuracy by iteratively optimizing the parallelly annotated corpora. Both source-to-target and target-to-source annotation transformations are performed in each training iteration, and the transformed corpora are used to provide better annotations for the parallelly annotated corpora of the next iteration; then the better parallelly annotated corpora will result in more accurate transformation classifiers, which will generate better transformed corpora in the new iteration. The predict-self reestimation is based on the following hypothesis, a better transformation result should be easier to be transformed back to the original form. The predict-self heuristic is also validated by Daumé III (2009) in unsupervised dependency parsing.

Experiments in Chinese word segmentation show that, the iterative training strategy together with predict-self reestimation brings significant improvement over the simple annotation transformation baseline. We perform optimized annotation transformation from the People's Daily (Yu et al., 2001) to the Penn Chinese Treebank 5.0 (CTB) (Xue et al., 2005), in order to improve the word segmenter with CTB annotation guideline. Compared to annotation adaptation, the optimized annotation transformation strategy leads to classifiers with significantly higher accuracy and several times faster processing on the same data sets. On CTB 5.0, it achieves an F-measure of 98.43%, significantly outperforms previous works although using a single classifier with only local features.

The rest of the paper is organized as follows. Section 2 describes the classification-based Chinese word segmentation method. Section 3 details the simple annotation transformation algorithm and the two optimization methods. After the introduction of related works in section 4, we give the experimental results on Chinese word segmentation in section 5.

2 Classification-Based Chinese Word Segmentation

Chinese word segmentation can be formalized as the problem of sequence labeling (Xue and Shen, 2003), where each character in the sentence is given a boundary tag denoting its position in a word. Following Ng and Low (2004), joint word segmentation and part-of-speech (POS) tagging can also be solved in a character classification approach by extending the boundary tags to include POS information. For word segmentation we adopt the 4 boundary tags of Ng and Low (2004), b, m, e and s, where b, m and e mean the beginning, the middle and the end of a word, and s indicates a single-character word. The word segmentation result can be generated by splitting the labeled character sequence into subsequences of pattern s or bm^*e, indicating single-character words or multi-character words, respectively.

Algorithm 1 Perceptron training algorithm.

```
1: Input: Training examples $(x_i, y_i)$
2: $\vec{\alpha} \leftarrow \mathbf{0}$
3: for $t \leftarrow 1 .. T$ do
4:     for $i \leftarrow 1 .. N$ do
5:         $z_i \leftarrow \mathrm{argmax}_{z \in \mathbf{GEN}(x_i)} \mathbf{\Phi}(x_i, z) \cdot \vec{\alpha}$
6:         if $z_i \neq y_i$ then
7:             $\vec{\alpha} \leftarrow \vec{\alpha} + \mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, z_i)$
8: Output: Parameters $\vec{\alpha}$
```

We choose the perceptron algorithm (Collins, 2002) to train the character classifier. It is an online training algorithm and has been successfully used in many NLP tasks, including POS tagging (Collins, 2002), parsing (Collins and Roark, 2004) and word segmentation (Zhang and Clark, 2007; Jiang et al., 2008; Zhang and Clark, 2010).

The training procedure learns a discriminative model mapping from the inputs $x \in X$ to the outputs $y \in Y$, where X is the set of sentences in the training corpus and Y is the set of corresponding labeled results. We use the function $\mathbf{GEN}(x)$ to enumerate the candidate results of an input x, and the function $\mathbf{\Phi}$ to map a training example $(x, y) \in X \times Y$ to a feature vector $\mathbf{\Phi}(x, y) \in R^d$. Given the character sequence x, the decoder finds the output $F(x)$ that maximizes the score function:

$$\begin{aligned} F(x) &= \underset{y \in \mathbf{GEN}(x)}{\mathrm{argmax}} \ \mathbf{S}(y|\vec{\alpha}, \mathbf{\Phi}, x) \\ &= \underset{y \in \mathbf{GEN}(x)}{\mathrm{argmax}} \ \mathbf{\Phi}(x, y) \cdot \vec{\alpha} \end{aligned} \tag{1}$$

Where $\vec{\alpha} \in R^d$ is the parameter vector (that is, the discriminative model) and $\mathbf{\Phi}(x, y) \cdot \vec{\alpha}$ is the inner product of $\mathbf{\Phi}(x, y)$ and $\vec{\alpha}$.

Algorithm 1 shows the perceptron algorithm for tuning the parameter $\vec{\alpha}$. The "averaged parameters"

Type	Feature Templates		
Unigram	C_{-2}	C_{-1}	C_0
	C_1	C_2	
Bigram	$C_{-2}C_{-1}$	$C_{-1}C_0$	C_0C_1
	C_1C_2	$C_{-1}C_1$	
Property	$Pu(C_0)$		
	$T(C_{-2})T(C_{-1})T(C_0)T(C_1)T(C_2)$		

Table 1: Feature templates for classification-based Chinese segmentation model.

technology (Collins, 2002) is used for better performance. The feature templates for the classifier is shown in Table 1. C_0 denotes the current character, while C_{-i}/C_i denote the ith character to the left/right of C_0. The function $Pu(\cdot)$ returns *true* for a punctuation character and *false* for others, the function $T(\cdot)$ classifies a character into four types: number, date, English letter and others.

3 Iterative and Predict-Self Annotation Transformation

This section first describes the technology of automatic annotation transformation, then introduces the two optimization strategies, iterative training and predict-self reestimation. Iterative training takes a global view, it conducts several rounds of bidirectional annotation transformations, and improve the transformation performance round by round. Predict-self reestimation takes a local view instead, it considers each training sentence, and improves the transformation performance by taking into account the predication result of the reverse transformation. The two strategies can be adopted jointly to obtain better transformation performance.

3.1 Automatic Annotation Transformation

Annotation adaptation can integrate the knowledge from two corpora with different underling annotation guidelines. First, a classifier (source classifier) is trained on the corpus (source corpus) with an annotation standard (source annotation) not desired, it is then used to classify the corpus (target corpus) with the annotation standard (target annotation) we want. Then a second classifier (transformation classifier [1]) is trained on the target corpus with

Type	Feature Templates		
Baseline	C_{-2}	C_{-1}	C_0
	C_1	C_2	
	$C_{-2}C_{-1}$	$C_{-1}C_0$	C_0C_1
	C_1C_2	$C_{-1}C_1$	
	$Pu(C_0)$		
	$T(C_{-2})T(C_{-1})T(C_0)T(C_1)T(C_2)$		
Guiding	α		
	$C_{-2}\circ\alpha$	$C_{-1}\circ\alpha$	$C_0\circ\alpha$
	$C_1\circ\alpha$	$C_2\circ\alpha$	
	$C_{-2}C_{-1}\circ\alpha$	$C_{-1}C_0\circ\alpha$	$C_0C_1\circ\alpha$
	$C_1C_2\circ\alpha$	$C_{-1}C_1\circ\alpha$	
	$Pu(C_0)\circ\alpha$		
	$T(C_{-2})T(C_{-1})T(C_0)T(C_1)T(C_2)\circ\alpha$		

Table 2: Feature templates for annotation transformation, where α is short for $\alpha(C_0)$, representing the source annotation of C_0.

the source classifier's classification result as guiding features. In decoding, a raw sentence is first decoded by the source classifier, and then inputted into the transformation classifier together with the annotations given by the source classifier, so as to obtain an improved classification result.

However, annotation adaptation has a drawback, it has to cascade two classifiers in decoding to integrate the knowledge in two corpora, thus seriously degrades the processing speed. This paper describes a variant of annotation adaptation, name annotation transformation, aiming at automatic transformation (rather than adaptation) between annotation standards of human-annotated corpora. In annotation transformation, a source classifier and a transformation classifier are trained in the same way as in annotation adaptation. The transformation classifier is used to process the source corpus, with the classification label derived from the segmented sentences as the guiding features, so as to relabel the source corpus with the target annotation guideline. By integrating the target corpus and the transformed source corpus for the training of the character classifier, improved classification accuracy can be achieved.

Both the source classifier and the transformation classifier are trained with the perceptron algorithm. The feature templates used for the source classifier are the same with those for the baseline

[1]It is called *target classifier* in (Jiang et al., 2009). We think that *transformation classifier* better reflects its role, the renaming also avoids name confusion in the optimized annotation transformation.

Algorithm 2 Baseline annotation transformation.

```
1: function AnnoTrans($\mathcal{C}_s$, $\mathcal{C}_t$)
2:     $\mathcal{M}_s \leftarrow$ Train($\mathcal{C}_s$)
3:     $\mathcal{C}_t^s \leftarrow$ Annotate($\mathcal{M}_s, \mathcal{C}_t$)
4:     $\mathcal{M}_{s\to t} \leftarrow$ TransTrain($\mathcal{C}_t^s, \mathcal{C}_t$)
5:     $\mathcal{C}_s^t \leftarrow$ TransAnnotate($\mathcal{M}_{s\to t}, \mathcal{C}_s$)
6:     $\mathcal{C}_*^t \leftarrow \mathcal{C}_s^t \cup \mathcal{C}_t$
7:     return $\mathcal{C}_*^t$
8: function Decode($\mathcal{M}$, $\mathbf{\Phi}$, $x$)
9:     return $\mathrm{argmax}_{y\in\mathbf{GEN}(x)}\, \mathbf{S}(y|\mathcal{M}, \mathbf{\Phi}, x)$
```

character classifier. The feature templates for the transformation classifier are the same with those in annotation adaptation, as listed in Table 2. Algorithm 2 shows the overall training algorithm for annotation transformation. $\mathcal{C}_s$ and $\mathcal{C}_t$ denote the source corpus and the target corpus; $\mathcal{M}_s$ and $\mathcal{M}_{s\to t}$ denote the source classifier and the transformation classifier; $\mathcal{C}_p^q$ denotes the p corpus relabeled in q annotation guideline, for example $\mathcal{C}_s^t$ is the source corpus transformed to target annotation guideline; Functions TRAIN and TRANSTRAIN both invoke the perceptron algorithm, yet with different feature sets; Functions ANNOTATE and TRANSANNOTATE call the function DECODE with different models (source/transformation classifiers), feature functions (without/with guiding features), and inputs (raw/source-annotated sentences).

The best training iterations for the functions TRAIN and TRANSTRAIN are determined on the developing sets of the source corpus and the target corpus, respectively. In the algorithm the parameters corresponding to developing sets are omitted for simplicity. Compared to the online knowledge integration methodology of annotation adaptation, annotation transformation leads to improved performance in an offline manner by integrating corpora before the training procedure. This manner could achieve processing several times as fast as the cascaded classifiers in annotation adaptation. In the following we will describe the two optimization strategies in details.

3.2 Iterative Training for Annotation Transformation

The training of annotation transformation is based on an auto-generated (rather than gold) parallelly annotated corpus, where the source annotation is pro-

Algorithm 3 Iterative annotation transformation.

```
1: function IterAnnoTrans($\mathcal{C}_s$, $\mathcal{C}_t$)
2:     $\mathcal{M}_s \leftarrow$ Train($\mathcal{C}_s$)
3:     $\mathcal{C}_t^s \leftarrow$ Annotate($\mathcal{M}_s, \mathcal{C}_t$)
4:     $\mathcal{M}_t \leftarrow$ Train($\mathcal{C}_t$)
5:     $\mathcal{C}_s^t \leftarrow$ Annotate($\mathcal{M}_t, \mathcal{C}_s$)
6:     repeat
7:         $\mathcal{M}_{s\to t} \leftarrow$ TransTrain($\mathcal{C}_t^s, \mathcal{C}_t$)
8:         $\mathcal{M}_{t\to s} \leftarrow$ TransTrain($\mathcal{C}_s^t, \mathcal{C}_s$)
9:         $\mathcal{C}_s^t \leftarrow$ TransAnnotate($\mathcal{M}_{s\to t}, \mathcal{C}_s$)
10:        $\mathcal{C}_t^s \leftarrow$ TransAnnotate($\mathcal{M}_{t\to s}, \mathcal{C}_t$)
11:        $\mathcal{C}_*^t \leftarrow \mathcal{C}_s^t \cup \mathcal{C}_t$
12:        $\mathcal{M}_* \leftarrow$ Train($\mathcal{C}_*^t$)
13:    until Eval($\mathcal{M}_*$) converges
14:    return $\mathcal{C}_*^t$
15: function Decode($\mathcal{M}$, $\mathbf{\Phi}$, $x$)
16:    return $\mathrm{argmax}_{y\in\mathbf{GEN}(x)}\, \mathbf{S}(y|\mathcal{M}, \mathbf{\Phi}, x)$
```

vided by the source classifier. Therefore, the performance of transformation training is correspondingly determined by the accuracy of the source classifier.

We propose an iterative training procedure to gradually improve the transformation accuracy by iteratively optimizing the parallelly annotated corpora. In each training iteration, both source-to-target and target-to-source annotation transformations are performed, and the transformed corpora are used to provide better annotations for the parallelly annotated corpora of the next iteration. Then in the new iteration, the better parallelly annotated corpora will result in more accurate transformation classifiers, so as to generate better transformed corpora.

Algorithm 3 shows the overall procedure of the iterative training method. The loop of lines 6-13 iteratively performs source-to-target and target-to-source annotation transformations. The source annotations of the parallelly annotated corpora, $\mathcal{C}_t^s$ and $\mathcal{C}_s^t$, are initialized by applying the source and target classifiers respectively on the target and source corpora (lines 2-5). In each training iteration, the transformation classifiers are trained on the current parallelly annotated corpora (lines 7-8), they are used to produce the transformed corpora (lines 9-10) which provide better annotations for the parallelly annotated corpora of the next iteration. The iterative training terminates when the performance of the classifier trained on the merged corpus $\mathcal{C}_s^t \cup \mathcal{C}_t$ converges.

The discriminative training of TRANSTRAIN predicts the target annotations with the guidance of source annotations. In the first iteration, the transformed corpora generated by the transformation classifiers are better than the initialized ones generated by the source and target classifiers, due to the assistance of the guiding features. In the following iterations, the transformed corpora provide better annotations for the parallelly annotated corpora of the subsequent iteration, the transformation accuracy will improve gradually along with optimization of the parallelly annotated corpora until convergence.

3.3 Predict-Self Reestimation for Annotation Transformation

The predict-self hypothesis is implicit in many unsupervised learning approaches, such as Markov random field. This methodology has also been successfully used by Daumé III (2009) in unsupervised dependency parsing. The basic idea of predict-self is that, if a prediction is a better candidate for an input, it can be easier converted back to the original input by a reverse procedure. If applied to the task of annotation transformation, predict-self indicates that a better transformation candidate following the target annotation guideline can be easier transformed back to the original form following the source annotation guideline.

The most intuitionistic strategy to introduce the predict-self methodology into annotation transformation is using a reversed annotation transformation procedure to filter out unreliable predictions of the previous transformation. In detail, a source-to-target annotation transformation is performed on the source annotated sentence to obtain a prediction that follows the target annotation guideline, then a second, target-to-source transformation is performed on this prediction result to check whether it can be transformed back to the previous source annotation. Transformation results failing in this reversal verification are discarded, so this strategy is named predict-self filtration.

A more precious strategy can be called predict-self reestimation. Instead of using the reversed transformation procedure for filtration, the reestimation strategy integrates the scores given by the source-to-target and target-to-source annotation transformation models when evaluating the transformation candidates. By properly tuning the relative weights of the two transformation directions, better transformation performance would be achieved. The scores of the two transformation models are weighted integrated in a log-linear manner:

$$\begin{aligned}\mathbf{S}^{+}(y|\mathcal{M}_{s\to t},\mathcal{M}_{t\to s},\mathbf{\Phi},x)\\ =(1-\lambda)\times\mathbf{S}(y|\mathcal{M}_{s\to t},\mathbf{\Phi},x)\\ +\lambda\times\mathbf{S}(x|\mathcal{M}_{t\to s},\mathbf{\Phi},y)\end{aligned} \tag{2}$$

The weight parameter λ is tuned on the developing set. To integrating the predict-self reestimation into the iterative transformation training, a reversed transformation model is introduced and the enhanced scoring function above is used when the function TRANSANNOTATE invokes the function DECODE.

4 Related Works

Researches focused on the automatic adaptation between different corpora can be roughly classified into two kinds, adaptation between different domains (with different statistical distribution) (Blitzer et al., 2006; Daumé III, 2007), and adaptation between different annotation guidelines (Jiang et al., 2009; Zhu et al., 2011). There are also some efforts that totally or partially resort to manual transformation rules, to conduct treebank conversion (Cahill and Mccarthy, 2002; Hockenmaier and Steedman, 2007; Clark and Curran, 2009), and word segmentation guideline transformation (Gao et al., 2004; Mi et al., 2008). This work focuses on the automatic transformation between annotation guidelines, and proposes better annotation transformation technologies to improve the transformation accuracy and the utilization rate of human-annotated knowledge.

The iterative training procedure proposed in this work shares some similarity with the co-training algorithm in parsing (Sarkar, 2001), where the training procedure lets two different models learn from each other during parsing the raw text. The key idea of co-training is utilize the complementarity of different parsing models to mine additional training data from raw text, while iterative training for annotation transformation emphasizes the iterative optimization of the parellelly annotated corpora used

Partition	**Sections**	**# of word**
CTB		
Training	$1-270$	0.47M
	$400-931$	
	$1001-1151$	
Developing	$301-325$	6.66K
Test	$271-300$	7.82K
PD		
Training	$02-06$	5.86M
Test	01	1.07M

Table 3: Data partitioning for CTB and PD.

	Test on ($F_1\%$)	
Train on	CTB	SPD
CTB	97.35	$86.65(\downarrow 10.70)$
SPD	$91.23(\downarrow 3.02)$	94.25

Table 4: Performance of the perceptron classifiers for Chinese word segmentation.

Model	**Time** (s)	**Accuracy** ($F_1\%$)
Merging	$\mathbf{1.33}$	93.79
Anno. Adapt.	4.39	97.67
Anno. Trans.	$\mathbf{1.33}$	$\mathbf{97.69}$
Baseline	1.21	97.35

Table 5: Comparison of the baseline annotation transformation, annotation adaptation and a simple corpus merging strategy.

to train the transformation models. The predict-self methodology is implicit in many unsupervised learning approaches, it has been successfully used by (Daumé III, 2009) in unsupervised dependency parsing. We adapt this idea to the scenario of annotation transformation to improve transformation accuracy.

In recent years many works have been devoted to the word segmentation task. For example, the introduction of global training or complicated features (Zhang and Clark, 2007; Zhang and Clark, 2010); the investigation of word structures (Li, 2011); the strategies of hybrid, joint or stacked modeling (Nakagawa and Uchimoto, 2007; Kruengkrai et al., 2009; Wang et al., 2010; Sun, 2011), and the semi-supervised and unsupervised technologies utilizing raw text (Zhao and Kit, 2008; Johnson and Goldwater, 2009; Mochihashi et al., 2009; Hewlett and Cohen, 2011). We estimate that the annotation transformation technologies can be adopted jointly with complicated features, system combination and semi-supervised/unsupervised technologies to further improve segmentation performance.

5 Experiments and Analysis

We perform annotation transformation from People's Daily (PD) (Yu et al., 2001) to Penn Chinese Treebank 5.0 (CTB) (Xue et al., 2005), following the same experimental setting as the annotation adaptation work (Jiang et al., 2009) for convenience of comparison. The two corpora are segmented following different segmentation guidelines and differ largely in quantity of data. CTB is smaller in size with about 0.5M words, while PD is much larger, containing nearly 6M words.

To approximate more general scenarios of annotation adaptation problems, we extract from PD a subset which is comparable to CTB in size. We randomly select $20,000$ sentences (0.45M words) from the PD training data as the new training set, and $1000/1000$ sentences from the PD test data as the new test/developing set. [2] We name the smaller version of PD as SPD. The balanced source corpus and target corpus also facilitate the investigation of annotation transformation.

5.1 Baseline Classifiers for Word Segmentation

We train the baseline perceptron classifiers described in section 2 on the training sets of SPD and CTB, using the developing sets to determine the best training iterations. The performance measurement indicators for word segmentation is balanced F-measure, $F = 2PR/(P + R)$, a function of Precision P and Recall R. where P is the percentage of words in segmentation result that are segmented correctly, and R is the percentage of correctly segmented words in the gold standard words.

Accuracies of the baseline classifiers are listed in Table 4. We also report the performance of the classifiers on the test sets of the opposite corpora. Experimental results are in line with our expectations. A classifier performs better in its corresponding test set, and performs significantly worse on a test set following a different annotation guideline.

[2]There are many extremely long sentences in original PD corpus, we split them into normal sentences according to period punctuations.

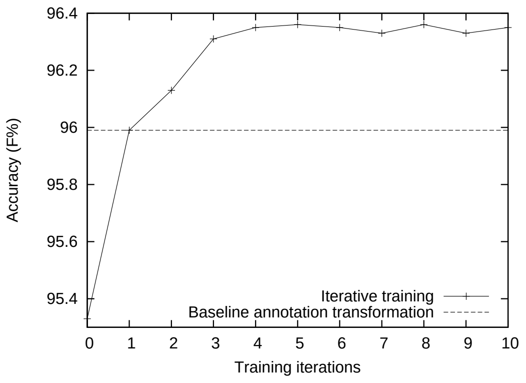

Figure 1: Learning curve of iterative training for annotation transformation.

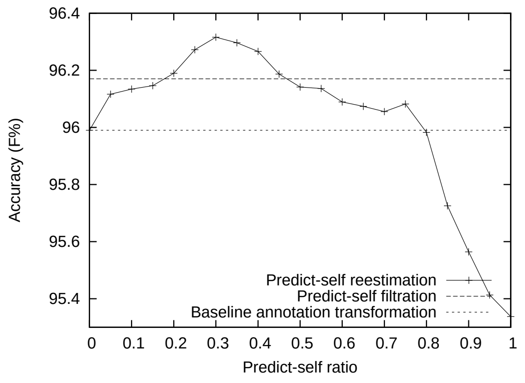

Figure 2: Performance of predict-self filtration and predict-self reestimation.

5.2 Annotation Transformation vs. Annotation Adaptation

Experiments of annotation transformation are conducted on the direction of SPD-to-CTB. The transformed corpus can be merged into the regular corpus, so as to train an enhanced classifier. As comparison, the cascaded model of annotation adaptation (Jiang et al., 2009) is faithfully implemented (yet using our feature representation) and tested on the same adaptation direction.

Table 5 shows the performances of the classifiers resulted by the baseline annotation transformation and annotation adaptation, as well as the classifier trained on the directly merged corpus. The time costs for decoding are also listed to facilitate the comparison of practicality. We find that the simple corpus merging strategy leads to dramatic decrease in accuracy, due to the different and incompatible annotation guidelines. The baseline annotation transformation method leads to a classifier with accuracy increment comparable to that of the annotation adaptation strategy, while consuming only one third of the decoding time.

5.3 Iterative Training with Predict-Self Reestimation

We adopt the iterative training strategy to the baseline annotation transformation model. The CTB developing set is used to determine the best training iteration for annotation transformation from SPD to CTB. After each iteration, we test the performance of the classifier trained on the merged corpus. Figure 1 shows the performance curve, with iterations

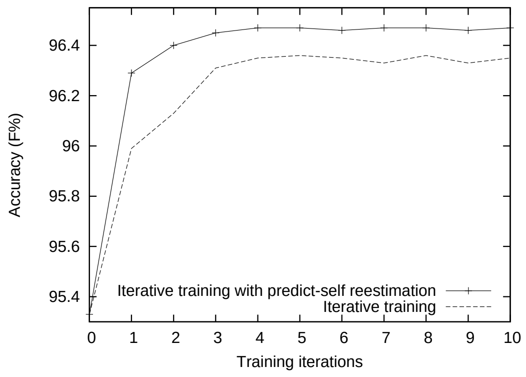

Figure 3: Learning curve of iterative training with predict-self reestimation for annotation transformation.

ranging from 1 to 10. The performance of the baseline annotation transformation model is naturally included in the curve (located at iteration 1). The curve shows that the performance of the classifier trained on the merged corpus consistently improves from iteration 2 to iteration 5.

Experimental results of predict-self filtration and predict-self reestimation are shown in Figure 2. The curve shows the performance of the predict-self reestimation according to a series of weight parameters, ranging from 0 to 1 with step 0.05. The point at $\lambda = 0$ shows the performance of the baseline annotation transformation strategy. The upper horizontal line shows the performance of predict-self filtration. We find that predict-self filtration brings slight improvement over the baseline, and predict-self reestimation outperforms the filtration strategy when λ falls in a proper range. An initial analysis on the experimental results of predict-self filtration

Model	**Time** (s)	**Accuracy** ($F_1\%$)
SPD → CTB		
Anno. Adapt.	4.39	97.67
Opt. Trans.	**1.33**	**97.97**
PD → CTB		
Anno. Adapt.	4.76	98.15
Opt. Trans.	**1.37**	**98.43**
Previous Works		
(Jiang et al., 2008)		97.85
(Kruengkrai et al., 2009)		97.87
(Zhang and Clark, 2010)		97.79
(Sun, 2011)		98.17

Table 6: The performance of the iterative annotation transformation with predict-self reestimation compared with annotation adaptation.

shows that, the filtration discards 5% of the training sentences and these discarded sentences contain nearly 10% of training words. It can be confirmed that the sentences discarded by predict-self filtration are much longer and more complicated. With a properly tuned weight, predict-self reestimation can make better use of the training data. The best F-measure improvement achieved over the annotation transformation baseline is 0.3 points, a little worse than that brought by iterative training.

Figure 3 shows the performance curve of iterative annotation transformation with predict-self reestimation. We find that the predict-self reestimation brings improvement to the iterative training at each iteration. The maximum performance is achieved at iteration 4. The corresponding model is evaluated on the test set of CTB, table 6 shows the experimental results. Compared to annotation adaptation, the optimized annotation transformation strategy leads to a classifier with significantly higher accuracy and several times faster processing. When using the whole PD as the source corpus, the final classifier [3] achieves an F-measure of 98.43%, significantly outperforms previous works although using a single classifier with only local features. Of course, the comparison between our system and previous works without using additional training data is unfair. This work aim to find another way to improve Chinese word segmentation, which focuses on the collection of more training data instead of making full use of a certain corpus. We believe that the performance can be further improved by adopting the advanced technologies of previous works, such as complicated features and model combination.

Considering the fact that today some corpora for word segmentation are really large (usually tens of thousands of sentences), it is necessary to obtain the latest CTB and investigate whether and how much does annotation transformation bring improvement to a much higher baseline. On the other hand, it is valuable to conduct experiments with more source-annotated training data, such as the SIGHAN dataset, to investigate the trend of improvement along with the increment of the additional annotated sentences. It is also valuable to evaluate the improved word segmenter on the out-of-domain datasets. However, currently most corpora for Chinese word segmentation do not explicitly distinguish the domains of their data sections, it makes such evaluations difficult to conduct.

6 Conclusion and Future Works

In this paper, we first describe an annotation transformation algorithm to automatically transform a human-annotated corpus from one annotation guideline to another. Then we propose two optimization strategies, iterative training and predict-self reestimation, to further improve the accuracy of annotation guideline transformation. On Chinese word segmentation, the optimized annotation transformation strategy leads to classifiers with obviously better performance and several times faster processing on the same datasets, compared to annotation adaptation. When adopting the whole PD as the source corpus, the final classifier significantly outperforms previous works on CTB 5.0, although using a single classifier with only local features.

As future works, we will investigate the acceleration of the iterative training and the weight parameter tuning, and extend the optimized annotation transformation strategy to joint Chinese word segmentation and POS tagging, parsing and other NLP tasks.

Acknowledgments

The authors were supported by National Natural Science Foundation of China, Contracts 90920004

[3]The predict-self reestimation ratio λ is fixed after the first training iteration for efficiency.

and 61100082, and 863 State Key Project No. 2011AA01A207. We are grateful to the anonymous reviewers for their thorough reviewing and valuable suggestions.

References

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of EMNLP*.

Aoife Cahill and Mairead Mccarthy. 2002. Automatic annotation of the penn treebank with lfg f-structure information. In *in Proceedings of the LREC Workshop*.

Stephen Clark and James R. Curran. 2009. Comparing the accuracy of ccg and penn treebank parsers. In *Proceedings of ACL-IJCNLP*.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of ACL 2004*.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*, pages 1–8, Philadelphia, USA.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *Proceedings of ACL*.

Hal Daumé III. 2009. Unsupervised search-based structured prediction. In *Proceedings of ICML*.

Jianfeng Gao, Andi Wu, Mu Li, Chang-Ning Huang, Hongqiao Li, Xinsong Xia, and Haowei Qin. 2004. Adaptive chinese word segmentation. In *Proceedings of ACL*.

Daniel Hewlett and Paul Cohen. 2011. Fully unsupervised word segmentation with bve and mdl. In *Proceedings of ACL*.

Julia Hockenmaier and Mark Steedman. 2007. Ccgbank: a corpus of ccg derivations and dependency structures extracted from the penn treebank. In *Computational Linguistics*, volume 33(3), pages 355–396.

Wenbin Jiang, Liang Huang, Yajuan Lv, and Qun Liu. 2008. A cascaded linear model for joint chinese word segmentation and part-of-speech tagging. In *Proceedings of ACL*.

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*.

Mark Johnson and Sharon Goldwater. 2009. Improving nonparameteric bayesian inference: experiments on unsupervised word segmentation with adaptor grammars. In *Proceedings of NAACL*.

Canasai Kruengkrai, Kiyotaka Uchimoto, Junichi 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 *Proceedings of ACL-IJCNLP*.

Zhongguo Li. 2011. Parsing the internal structure of words: A new paradigm for chineseword segmentation. In *Proceedings of ACL*.

Haitao Mi, Deyi Xiong, and Qun Liu. 2008. Research on strategy of integrating chinese lexical analysis and parser. *In Journal of Chinese Information Processing*.

Daichi Mochihashi, Takeshi Yamada, and Naonori Ueda. 2009. Bayesian unsupervised word segmentation with nested pitman-yor language modeling. In *Proceedings of ACL-IJCNLP*.

Tetsuji Nakagawa and Kiyotaka Uchimoto. 2007. A hybrid approach to word segmentation and pos tagging. In *Proceedings of ACL*.

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 *Proceedings of EMNLP*.

Anoop Sarkar. 2001. Applying co-training methods to statistical parsing. In *Proceedings of NAACL*.

Weiwei Sun. 2011. A stacked sub-word model for joint chinese word segmentation and part-of-speech tagging. In *Proceedings of ACL*.

Kun Wang, Chengqing Zong, and Keh-Yih Su. 2010. A character-based joint model for chinese word segmentation. In *Proceedings of COLING*.

Nianwen Xue and Libin Shen. 2003. Chinese word segmentation as lmr tagging. In *Proceedings of SIGHAN Workshop*.

Nianwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. In *Natural Language Engineering*.

Shiwen Yu, Jianming Lu, Xuefeng Zhu, Huiming Duan, Shiyong Kang, Honglin Sun, Hui Wang, Qiang Zhao, and Weidong Zhan. 2001. Processing norms of modern chinese corpus. Technical report.

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

Yue Zhang and Stephen Clark. 2010. A fast decoder for joint word segmentation and pos-tagging using a single discriminative model. In *Proceedings of EMNLP*.

Hai Zhao and Chunyu Kit. 2008. Unsupervised segmentation helps supervised learning of character tagging for word segmentation and named entity recognition. In *Proceedings of SIGHAN Workshop*.

Muhua Zhu, Jingbo Zhu, and Minghan Hu. 2011. Better automatic treebank conversion using a feature-based approach. In *Proceedings of ACL*.