Automatic Adaptation of Annotations

Wenbin Jiang*
Chinese Academy of Sciences

Yajuan Lü*
Chinese Academy of Sciences

Liang Huang**
Queens College and Graduate Center,
The City University of New York

Qun Liu*†
Dublin City University
Chinese Academy of Sciences

Manually annotated corpora are indispensable resources, yet for many annotation tasks, such as the creation of treebanks, there exist multiple corpora with different and incompatible annotation guidelines. This leads to an inefficient use of human expertise, but it could be remedied by integrating knowledge across corpora with different annotation guidelines. In this article we describe the problem of annotation adaptation and the intrinsic principles of the solutions, and present a series of successively enhanced models that can automatically adapt the divergence between different annotation formats.

We evaluate our algorithms on the tasks of Chinese word segmentation and dependency parsing. For word segmentation, where there are no universal segmentation guidelines because of the lack of morphology in Chinese, we perform annotation adaptation from the much larger People's Daily corpus to the smaller but more popular Penn Chinese Treebank. For dependency parsing, we perform annotation adaptation from the Penn Chinese Treebank to a semantics-oriented Dependency Treebank, which is annotated using significantly different annotation guidelines. In both experiments, automatic annotation adaptation brings significant improvement, achieving state-of-the-art performance despite the use of purely local features in training.

* Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, No. 6 Kexueyuan South Road, Haidian District, P.O. Box 2704, Beijing 100190, China. E-mail: {jiangwenbin, liuqun, lvyajuan}@ict.ac.cn.
** Department of Computer Science, Queens College / CUNY, 65-30 Kissena Blvd., Queens, NY 11367. E-mail: liang.huang.sh@gmail.com.
† Centre for Next Generation Localisation, Faculty of Engineering and Computing, Dublin City University. E-mail: qliu@computing.dcu.ie.

Submission received: 24 April 2013; revised version received: 6 March 2014; accepted for publication: 18 April 2014.

doi:10.1162/COLI_a_00210

1. Introduction

Much of statistical NLP research relies on some sorts of manually annotated corpora to train models, but annotated resources are extremely expensive to build, especially on a large scale. The creation of treebanks is a prime example (Marcus, Santorini, and Marcinkiewicz 1993). However, the linguistic theories motivating these annotation efforts are often heavily debated, and as a result there often exist multiple corpora for the same task with vastly different and incompatible annotation philosophies. For example, there are several treebanks for English, including the Chomskian-style Penn Treebank (Marcus, Santorini, and Marcinkiewicz 1993), the HPSG LinGo Redwoods Treebank (Oepen et al. 2002), and a smaller dependency treebank (Buchholz and Marsi 2006). From the perspective of resource accumulation, it seems a waste in human efforts.[1]

A second, related problem is that the raw texts are also drawn from different domains, which for the above example range from financial news (Penn Treebank/*Wall Street Journal*) to transcribed dialog (LinGo). It would be nice if a system could be automatically ported from one set of guidelines and/or domain to another, in order to exploit a much larger data set. The second problem, domain adaptation, is very well studied (e.g., Blitzer, McDonald, & Pereira 2006; Daumé III 2007). This work focuses on the widely existing and equally important problem, annotation adaptation, in order to adapt the divergence between different annotation guidelines and integrate linguistic knowledge in corpora with incongruent annotation formats.

In this article, we describe the problem of annotation adaptation and the intrinsic principles of the solutions, and present a series of successively improved concrete models, the goal being to transfer the annotations of a corpus (**source corpus**) to the annotation format of another corpus (**target corpus**). The **transfer classifier** is the fundamental component for annotation adaptation algorithms. It learns correspondence regularities between annotation guidelines from a **parallel annotated corpus**, which has two kinds of annotations for the same data. In the simplest model (Model 1), the **source classifier** trained on the source corpus gives its predications to the transfer classifier trained on the parallel annotated corpus, so as to integrate the knowledge in the two corpora. In a variant of the simplest model (Model 2), the transfer classifier is used to transform the annotations in the source corpus into the annotation format of the target corpus; then the transformed source corpus and the target corpus are merged in order to train a more accurate classifier. Based on the second model, we finally develop an optimized model (Model 3), where two optimization strategies, iterative training and predict-self re-estimation, are integrated to further improve the efficiency of annotation adaptation.

We experiment on Chinese word segmentation and dependency parsing to test the efficacy of our methods. For word segmentation, the problem of incompatible annotation guidelines is one of the most glaring: No segmentation guideline has been widely accepted due to the lack of a clear definition of Chinese word morphology. For dependency parsing there also exist multiple disparate annotation guidelines. For

1 Different annotated corpora for the same task facilitate the comparison of linguistic theories. From this perspective, having multiple standards is not necessarily a waste but rather a blessing, because it is a necessary phase in coming to a consensus if there is one.

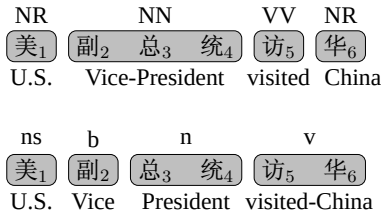


Figure 1
Incompatible word segmentation and POS tagging guidelines between CTB (upper) and PD (below).

example, the dependency relations extracted from a constituency treebank follow syntactic principles, whereas the semantic dependency treebank is annotated in a semantic perspective.

The two corpora for word segmentation are the much larger People's Daily corpus (PD) (5.86M words) (Yu et al. 2001) and the smaller but more popular Penn Chinese Treebank (CTB) (0.47M words) (Xue et al. 2005). They utilize very different segmentation guidelines; for example, as shown in Figure 1, PD breaks *Vice-President* into two words and combines the phrase *visited-China* as a compound, compared with the segmentation following the CTB annotation guideline. It is preferable to transfer knowledge from PD to CTB because the latter also annotates tree structures, which are useful for downstream applications like parsing, summarization, and machine translation, yet it is much smaller in size. For dependency parsing, we use the dependency treebank (DCTB) extracted from CTB according to the rules of Yamada and Matsumoto (2003), and the Semantic Dependency Treebank (SDT) built on a small part of the CTB text (Che et al. 2012). Compared with the automatically extracted dependencies in DCTB, semantic dependencies in SDT reveal semantic relationships between words, rather than the syntactic relationships in syntactic dependencies. Figure 2 shows an example. Experiments on both word segmentation and dependency parsing show that annotation adaptation results in significant improvement over the baselines, and achieves the state-of-the-art with only local features.

The rest of the article is organized as follows. Section 2 gives a description of the problem of annotation adaptation. Section 3 briefly introduces the tasks of word segmentation and dependency parsing as well as their state-of-the-art models. In Section 4 we first describe the transfer classifier that indicates the

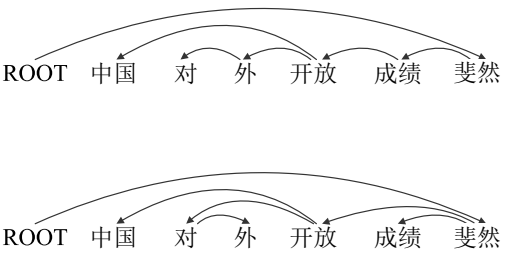


Figure 2
Incompatible dependency guidelines between SDT (top) and DCTB (bottom).

intrinsic principles of annotation adaptation, and then depict the three successively enhanced models for automatic adaptation of annotations. After the description of experimental results in Section 5 and the discussion of application scenarios in Section 6, we give a brief review of related work in Section 7, drawing conclusions in Section 8.

2. Automatic Annotation Adaptation

We define **annotation adaptation** as a task aimed at automatically adapting the divergence between different annotation guidelines. Statistical models can be designed to learn the relevance of two annotation guidelines in order to transform a corpus from one annotation guideline to another. From this point of view, annotation adaptation can be seen as a special case of transfer learning. Through annotation adaptation, the linguistic knowledge in different corpora is integrated, resulting in enhanced NLP systems without complicated models and features.

Much research has considered the problem of domain adaptation (Blitzer, McDonald, and Pereira 2006; Daumé III 2007), which also can be seen as a special case of transfer learning. It aims to adapt models trained in one domain (e.g., chemistry) to work well in other domains (e.g., medicine). Despite superficial similarities between domain adaptation and annotation adaptation, we argue that the underlying problems are quite different. Domain adaptation assumes that the labeling guidelines are preserved between the two domains—for example, an adjective is always labeled as JJ regardless of whether it is from the *Wall Street Journal* (WSJ) or a biomedical text, and only the *distributions* are different—for example, the word *control* is most likely a verb in WSJ but often a noun in biomedical texts (as in *control experiment*). Annotation adaptation, however, tackles the problem where the guideline itself is changed, for example, one treebank might distinguish between transitive and intransitive verbs, while merging the different noun types (NN, NNS, etc.), or one treebank (PTB) might be much flatter than the other (LinGo), not to mention the fundamental disparities between their underlying linguistic representations (CFG vs. HPSG).

A more formal description will allow us to make these claims more precise. Let X be the data and Y be the annotation. Annotation adaptation can be understood as a change of $P(Y)$ due to the change in annotation guidelines while $P(X)$ remains constant. Through annotation adaptation, we want to change the annotations of the data from one guideline to another, leaving the data itself unchanged. However, in domain adaptation, $P(X)$ changes, but $P(Y)$ is *assumed* to be constant. The word *assumed* means that the distributions $P(Y, X)$ and $P(Y|X)$ are actually changed because $P(X)$ is changed. Domain adaptation aims to make the model better adapt to a *different* domain with the *same* annotation guidelines.

According to this analysis, annotation adaptation seems more motivated from a linguistic (rather than statistical) point of view, and tackles a serious problem fundamentally different from domain adaptation, which is also a serious problem (often leading to >10% loss in accuracy). More interestingly, annotation adaptation, without any assumptions about distributions, can be simultaneously applied to both domain and annotation adaptation problems, which is very appealing in practice because the latter problem often implies the former.

3. Case Studies: Word Segmentation and Dependency Parsing

3.1 Word Segmentation and Character Classification Method

In many Asian languages there are no explicit word boundaries, thus word segmentation is a fundamental task for the processing and understanding of these languages. Given a sentence as a sequence of n characters:

$$x = x_1\ x_2\ ..\ x_n$$

where x_i is a character, word segmentation aims to split the sequence into $m(\leq n)$ words:

$$x_{1:e_1}\ x_{e_1+1:e_2}\ ..\ x_{e_{m-1}+1:e_m}$$

where each subsequence $x_{i:j}$ indicates a Chinese word spanning from characters x_i to x_j.

Word segmentation can be formalized as a sequence labeling problem (Xue and Shen 2003), where each character in the sentence is given a boundary tag representing its position in a word. Following Ng and Low (2004), joint word segmentation and part-of-speech (POS) tagging can also be solved using a character classification approach by extending boundary tags to include POS information. For word segmentation we adopt the four 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, respectively, 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.

Given the character sequence x, the decoder finds the output $\tilde{y}$ that maximizes the score function:

$$\begin{aligned} \tilde{y} &= \operatorname*{argmax}_y \mathbf{f}(x, y) \cdot \mathbf{w} \\ &= \operatorname*{argmax}_y \sum_{x_i \in x, y_i \in y} \mathbf{f}(x_i, y_i) \cdot \mathbf{w} \end{aligned} \tag{1}$$

Where function $\mathbf{f}$ maps (x, y) into a feature vector, $\mathbf{w}$ is the parameter vector generated by the training algorithm, and $\mathbf{f}(x, y) \cdot \mathbf{w}$ is the inner product of $\mathbf{f}(x, y)$ and $\mathbf{w}$. The score of the sentence is further factorized into each character, where y_i is the character classification label of character x_i.

The training procedure of perceptron learns a discriminative model mapping from the inputs x to the outputs y. Algorithm 1 shows the perceptron algorithm for tuning the parameter $\mathbf{w}$. The "averaged parameters" technology (Collins 2002) is used for better performance. The feature templates of the classifier is shown in Table 1. 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*, corresponding to function values 1, 2, 3, and 4, respectively.

3.2 Dependency Parsing and Spanning Tree Method

Dependency parsing aims to link each word to its arguments so as to form a directed graph spanning the whole sentence. Normally, the directed graph is restricted to a

Algorithm 1 Perceptron training algorithm.

Input: Training set $\mathbb{C}$
$\mathbf{w} \leftarrow \mathbf{0}$
for $t \leftarrow 1 \,..\, T$ **do** ▷ T iterations
  for $(x, y) \in \mathbb{C}$ **do**
    $\tilde{z} \leftarrow \text{argmax}_z \, \mathbf{f}(x, z) \cdot \mathbf{w}$
    if $\tilde{z} \neq y$ **then**
      $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{f}(x, y) - \mathbf{f}(x, \tilde{z})$ ▷ update the parameters
    end if
  end for
end for
Output: Parameters $\mathbf{w}$

dependency tree where each word depends on exactly one parent, and all words find their parents. Given a sentence as a sequence n words:

$$x = x_1 \; x_2 \; .. \; x_n$$

dependency parsing finds a dependency tree y, where $(i, j) \in y$ is an edge from the head word x_i to the modifier word x_j. The root $r \in x$ in the tree y has no head word, and each of the other words, $j (j \in x \; \textbf{and} \; j \neq r)$, depends on a head word $i (i \in x \; \textbf{and} \; i \neq j)$.

For many languages, the dependency structures are supposed to be projective. If x_j is dependent on x_i, then all the words between i and j must be directly or indirectly dependent on x_i. Therefore, if we put the words in their linear order, preceded by the root, all edges can be drawn above the words without crossing. We follow this constraint because the dependency treebanks in our experiments are projective.

Following the edge-based factorization method (Eisner 1996), the score of a dependency tree can be factorized into the dependency edges in the tree. The spanning

Table 1
Feature templates and instances for character classification-based word segmentation model. C_0 denotes the current character, and C_{-i}/C_i denote the ith character to the left/right of C_0. Suppose we are considering the third character "总" in "美-副-总-统-访-华".

Type	Templates	Instances
n-gram	C_{-2}	C_{-2}=美
	C_{-1}	C_{-1}=副
	C_0	C_0=总
	C_1	C_1=统
	C_2	C_2=访
	$C_{-2}C_{-1}$	$C_{-2}C_{-1}$=美副
	$C_{-1}C_0$	$C_{-1}C_0$=副总
	C_0C_1	C_0C_1=总统
	C_1C_2	C_1C_2=统访
	$C_{-1}C_1$	$C_{-1}C_1$=副统
function	$Pu(C_0)$	$Pu(C_0)$=true
	$T(C_{-2:2})$	$T(C_{-2:2})$= 44444

tree method (McDonald, Crammer, and Pereira 2005) factorizes the score of the tree as the sum of the scores of all its edges, and the score of an edge is defined as the inner product of the feature vector $\mathbf{f}$ and the weight vector $\mathbf{w}$. Given a sentence x, the parsing procedure searches for the candidate dependency tree with the maximum score:

$$\begin{aligned} \tilde{y} &= \operatorname*{argmax}_{y} \mathbf{f}(x, y) \cdot \mathbf{w} \\ &= \operatorname*{argmax}_{y} \sum_{(i,j) \in y} \mathbf{f}(i, j) \cdot \mathbf{w} \end{aligned} \tag{2}$$

The averaged perceptron algorithm is used again to train the parameter vector. A bottom–up dynamic programming algorithm is designed to search for the candidate parse with the maximum score as shown in Algorithm 2, where $\mathbf{V}[i, j]$ contains the candidate dependency fragments of the span $[i, j]$. The feature templates are similar to those of the first-ordered MST model (McDonald, Crammer, and Pereira 2005). Each feature is composed of some words and POS tags surround word i and/or word j, as well as an optional distance representation between the two words. Table 2 shows the feature templates without distance representations.

4. Models for Automatic Annotation Adaptation

In this section, we present a series of discriminative learning algorithms for the automatic adaptation of annotation guidelines. To facilitate the description of the algorithms, several shortened forms are adopted for convenience of description. We use **source corpus** to denote the corpus with the annotation guideline that we do not require, which is of course the source side of adaptation, and **target corpus** denotes the corpus with the desired guideline. Correspondingly, the annotation guidelines of the two corpora are denoted as **source guideline** and **target guideline**, and the classifiers following the two annotation guidelines are respectively named as **source classifier** and **target classifier**. Given a **parallel annotated corpus**, that is, a corpus labeled with two annotation

Algorithm 2 Dependency parsing algorithm.

```
1:  Input: sentence x to be parsed
2:  for [i, j] ⊆ [1, |x|] in topological order do
3:      buf ← ∅
4:      for k ← i..j − 1 do                                  ▷ all partitions
5:          for l ∈ V[i, k] and r ∈ V[k + 1, j] do
6:              insert DERIV(l, r) into buf
7:              insert DERIV(r, l) into buf
8:          end for
9:      end for
10:     V[i, j] ← best K in buf
11: end for
12: Output: the best of V[1, |x|]
13: function DERIV(p, c)
14:     return p ∪ c ∪ {(p · root, c · root)}                ▷ new derivation
15: end function
```

Table 2
Feature templates and instances for dependency parsing. W_i/P_i denotes the word/POS of the hypothetic head, W_j/P_j denotes the word/POS of the hypothetic modifier, and $P_{x+/-1}$ denotes the POS to the right/left of the x-th token. Suppose we are considering the words "外" ($i = 2$) and "开放" ($j = 3$) in "中国/NR-对/P-外/NN-开放/VV-成绩/NN-斐然/VV".

Type	Templates	Instances
unigram	W_iP_i	W_iP_i=外∘NN
	W_i	W_i=外
	P_i	P_i=NN
	W_jP_j	W_jP_j=开放∘VV
	W_j	W_j=开放
	P_j	P_j=VV
bigram	$W_iP_iW_jP_j$	$W_iP_iW_jP_j$=外∘NN∘开放∘VV
	$W_iW_jP_j$	$W_iW_jP_j$=外∘开放∘VV
	$P_iW_jP_j$	$P_iW_jP_j$=NN∘开放∘VV
	$W_iP_iW_j$	$W_iP_iW_j$=外∘NN∘开放
	$W_iP_iP_j$	$W_iP_iP_j$=外∘NN∘VV
	W_iW_j	W_iW_j=外∘开放
	W_iP_j	W_iP_j=外∘VV
	P_iW_j	P_iW_j=NN∘开放
	P_iP_j	P_iP_j=NN∘VV
context	$P_iP_{i+1}P_{j-1}P_j$	$P_iP_{i+1}P_{j-1}P_j$=NN∘VV∘NN∘VV
	$P_{i-1}P_iP_{j-1}P_j$	$P_{i-1}P_iP_{j-1}P_j$=P∘NN∘NN∘VV
	$P_iP_{i+1}P_jP_{j+1}$	$P_iP_{i+1}P_jP_{j+1}$=NN∘VV∘VV∘NN
	$P_{i-1}P_iP_jP_{j+1}$	$P_{i-1}P_iP_jP_{j+1}$=P∘NN∘VV∘NN
	$P_{i-1}P_iP_{j-1}$	$P_{i-1}P_iP_{j-1}$=P∘NN∘NN
	$P_{i-1}P_iP_{j+1}$	$P_{i-1}P_iP_{j+1}$=P∘NN∘NN
	$P_iP_{i+1}P_{j-1}$	$P_iP_{i+1}P_{j-1}$=NN∘VV∘NN
	$P_iP_{i+1}P_{j+1}$	$P_iP_{i+1}P_{j+1}$=NN∘VV∘NN
	$P_{i-1}P_{j-1}P_j$	$P_{i-1}P_{j-1}P_j$=P∘NN∘VV
	$P_{i-1}P_jP_{j+1}$	$P_{i-1}P_jP_{j+1}$=P∘VV∘NN
	$P_{i+1}P_{j-1}P_j$	$P_{i+1}P_{j-1}P_j$=VV∘NN∘VV
	$P_{i+1}P_jP_{j+1}$	$P_{i+1}P_jP_{j+1}$=VV∘VV∘NN
	$P_iP_{j-1}P_j$	$P_iP_{j-1}P_j$=NN∘NN∘VV
	$P_iP_jP_{j+1}$	$P_iP_jP_{j+1}$=NN∘VV∘NN
	$P_{i-1}P_iP_j$	$P_{i-1}P_iP_j$=P∘NN∘VV
	$P_iP_{i+1}P_j$	$P_iP_{i+1}P_j$=NN∘VV∘VV

guidelines, a **transfer classifier** can be trained to capture the regularity of transformation from the **source annotation** to the **target annotation**. The classifiers mentioned here are normal discriminative classifiers that take a set of features as input and give a classification label as output. For the POS tagging problem, the classification label is a POS tag, and for the parsing task, the classification label is a dependency edge, a constituency span, or a shift-reduce action.

The parallel annotated corpus is the knowledge source of annotation adaptation. The annotation quality and data size of the parallel annotated corpus determine the accuracy of the transfer classifier. Such a corpus is difficult to build manually, although we can generate a noisy one automatically from the source corpus and the target corpus. For example, we can apply the source classifier on the target corpus, thus generating

a parallel annotated corpus with noisy source annotations and accurate target annotations. The training procedure of the transfer classifier predicts the target annotations with guiding features extracted from the source annotations. This approach can alleviate the effect of the noise in source annotations, and learn annotation adaptation regularities accurately. By reducing the noise in the automatically generated parallel annotated corpus, a higher accuracy of annotation adaptation can be achieved.

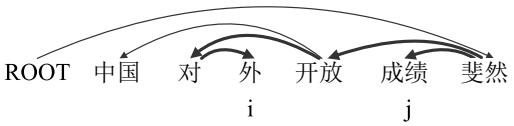
In the following sections, we first describe the transfer classifier that reveals the intrinsic principles of annotation adaptation, then describe a series of successively enhanced models that are developed from our previous investigation (Jiang, Huang, and Liu 2009; Jiang et al. 2012). In the simplest model (Model 1), two classifiers, a source classifier and a transfer classifier, are used in a cascade. The classification results of the lower source classifier provide additional guiding features to the upper transfer classifier, yielding an improved classification result. A variant of the first model (Model 2) uses the transfer classifier to transform the source corpus from source guideline to target guideline first, then merges the transformed source corpus into the target corpus in order to train an improved target classifier on the enlarged corpus. An optimized model (Model 3) is further proposed based on Model 2. Two optimization strategies, iterative training and predict-self re-estimation, are adopted to improve the efficiency of annotation adaptation, in order to fully utilize the knowledge in heterogeneous corpora.

4.1 Transfer Classifier

In order to learn the regularity of the adaptation from one annotation guideline to another, a parallel annotated corpus is needed to train the transfer classifier. The parallel annotated corpus is a corpus with two different annotation guidelines, the source guideline and the target guideline. With the target annotation labels as learning objectives and the source annotation labels as guiding information, the transfer classifier learns the statistical regularity of the adaptation from the source annotations to the target annotations.

The training procedure of the transfer classifier is analogous to the training of a normal classifier except for the introduction of additional guiding features. For word segmentation, the most intuitive guiding feature is the source annotation label itself. For dependency parsing, an effective guiding feature is the dependency path between the hypothetic head and modifier, as shown in Figure 3. However, our effort is not limited to this, and more special features are introduced: A classification label or dependency path is attached to each feature of the baseline classifier to generate combined guiding features. This is similar to the feature design in discriminative dependency parsing (McDonald, Crammer, and Pereira 2005; McDonald and Pereira 2006), where the basic features, composed of words and POSs in the context, are also conjoined with link direction and distance in order to generate more special features.

Table 3 shows an example of guide features (as well as baseline features) for word segmentation, where "α = B" indicates that the source classification label of the current character is *B*, demarcating the beginning of a word. The combination strategy derives a series of specific features, helping the transfer classifier to produce more precise classifications. The parameter-tuning procedure of the transfer classifier will automatically learn the regularity of using the source annotations to guide the classification decision. In decoding, if the current character shares some basic features in Table 3 and it is classified as *B* in the source annotation, then the transfer classifier will probably classify it as *M*.



α(外,成绩) = up-up-up-down

Figure 3
The guiding feature for dependency parsing, where $\alpha(i, j)$ denotes the dependency path between i and j. In this example, j is a son of the great-grandfather of i.

In addition, the original features used in the normal classifier are also used in order to leverage the knowledge from the target annotation of the parallel annotated corpus, and the training procedure of the transfer classifier also learns the relative weights between the guiding features and the original features. Therefore, the knowledge from both the source annotation and the target annotation are automatically integrated together, and higher and more stable prediction accuracy can be achieved.

Table 3
Feature templates and corresponding instances for the transfer classifier. Suppose we are considering the third character "总" in "美-副-总-统-访-华". $\alpha = B$ indicates that the classification label given by the source classifier is B.

Type	Templates	Instances
Original	C_{-2}	C_{-2}=美
	C_{-1}	C_{-1}=副
	C_0	C_0=总
	C_1	C_1=统
	C_2	C_2=访
	$C_{-2}C_{-1}$	$C_{-2}C_{-1}$=美副
	$C_{-1}C_0$	$C_{-1}C_0$=副总
	C_0C_1	C_0C_1=总统
	C_1C_2	C_1C_2=统访
	$C_{-1}C_1$	$C_{-1}C_1$=副统
	$Pu(C_0)$	$Pu(C_0)$=true
	$T(C_{-2:2})$	$T(C_{-2:2})$= 44444
Guiding	α	α=B
	$\alpha \circ C_{-2}$	α=B $\circ$ C_{-2}=美
	$\alpha \circ C_{-1}$	α=B $\circ$ C_{-1}=副
	$\alpha \circ C_0$	α=B $\circ$ C_0=总
	$\alpha \circ C_1$	α=B $\circ$ C_1=统
	$\alpha \circ C_2$	α=B $\circ$ C_2=访
	$\alpha \circ C_{-2}C_{-1}$	α=B $\circ$ $C_{-2}C_{-1}$=美副
	$\alpha \circ C_{-1}C_0$	α=B $\circ$ $C_{-1}C_0$=副总
	$\alpha \circ C_0C_1$	α=B $\circ$ C_0C_1=总统
	$\alpha \circ C_1C_2$	α=B $\circ$ C_1C_2=统访
	$\alpha \circ C_{-1}C_1$	α=B $\circ$ $C_{-1}C_1$=副统
	$\alpha \circ Pu(C_0)$	α=B $\circ$ $Pu(C_0)$=true
	$\alpha \circ T(C_{-2:2})$	α=B $\circ$ $T(C_{-2:2})$= 44444

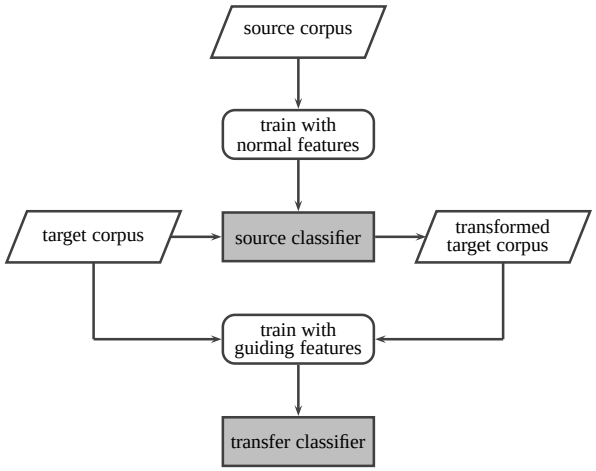


Figure 4
The pipeline for training of Model 1.

4.2 Model 1

The most intuitive model for annotation adaptation uses two cascaded classifiers, the source classifier and the transfer classifier, to integrate the knowledge in corpora with different annotation guidelines. In the training procedure, a source classifier is trained on the source corpus and is used to process the target corpus, generating a parallel annotated corpus (albeit a noisy one). Then, the transfer classifier is trained on the parallel annotated corpus,with the target annotations as the classification labels, and the source annotation as guiding information. Figure 4 depicts the training pipeline. The best training iterations for the source classifier and the transfer classifier are determined on the development sets of the source corpus and target corpus.

In the decoding procedure, a sequence of characters (for word segmentation) or words (for dependency parsing) is input into the source classifier to obtain a classification result under the source guideline; then it is input into the transfer classifier with this classification result as the guiding information to get the final result following the target guideline. This coincides with the stacking method for combining dependency parsers (Martins et al. 2008; Nivre and McDonald 2008), and is also similar to the Pred baseline for domain adaptation in Daumé et al. (Daumé III and Marcu 2006; Daumé III 2007). Figure 5 shows the pipeline for decoding.

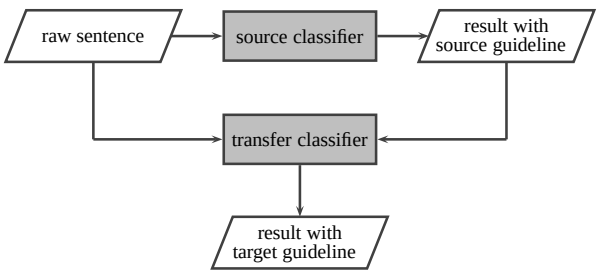


Figure 5
The pipeline for decoding of Model 1.

4.3 Model 2

The previous model has a drawback: It has to cascade two classifiers in decoding to integrate the knowledge in two corpora, which seriously degrades the processing speed. Here we describe a variant of the previous model, aiming at automatic *transformation* (rather than *integration* as in Model 1) between annotation guidelines of human-annotated corpora. The source classifier and the transfer classifier are trained in the same way as in the previous model. The transfer classifier is used to process the source corpus, with the source annotations as guiding information, so as to relabel the source corpus with the target annotation guideline. By merging the target corpus and the transformed source corpus for the training of the final classifier, improved classification accuracy can be achieved.

From this point on we describe the pipelines of annotation transformation in pseudo-codes for simplicity and convenience of extensions. Algorithm 3 shows the overall training algorithm for the variant model. $\mathcal{C}_s$ 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 transfer classifier. $\mathcal{C}_p^q$ denotes the p corpus relabeled in q annotation guideline; for example, $\mathcal{C}_s^t$ is a corpus that labels the text of the source corpus with the target guideline. Functions TRAIN and TRANSTRAIN train the source classifier and the transfer classifier, respectively, both invoking the perceptron algorithm, but with different feature sets. Functions ANNOTATE and TRANSANNOTATE call the function DECODE with different models (source/transfer classifiers), feature functions (without/with guiding features), and inputs (raw/source-annotated sentences).

In the algorithm the parameters corresponding to development sets are omitted for simplicity. Compared to the online knowledge integration methodology of the previous model, annotation transformation leads to improved performance in an offline manner by integrating corpora before the training procedure. This approach enables processing speeds several times faster than the cascaded classifiers in the previous model. It also has another advantage in that we can integrate the knowledge in more than two corpora without slowing down the processing of the final classifier.

4.4 Model 3

The training of the transfer classifier is based on an automatically generated (rather than a gold standard) parallel annotated corpus, where the source annotations are

Algorithm 3 Baseline annotation adaptation.

function ANNOTRANS($\mathcal{C}_s, \mathcal{C}_t$)
  $\mathcal{M}_s \leftarrow$ TRAIN($\mathcal{C}_s$) ▷ source classifier
  $\mathcal{C}_t^s \leftarrow$ ANNOTATE($\mathcal{M}_s, \mathcal{C}_t$)
  $\mathcal{M}_{s\to t} \leftarrow$ TRANSTRAIN($\mathcal{C}_t^s, \mathcal{C}_t$) ▷ transfer classifier
  $\mathcal{C}_s^t \leftarrow$ TRANSANNOTATE($\mathcal{M}_{s\to t}, \mathcal{C}_s$)
  $\mathcal{C}_*^t \leftarrow \mathcal{C}_s^t \cup \mathcal{C}_t$ ▷ integrated corpus with target guideline
  return $\mathcal{C}_*^t$
end function
function DECODE($\mathcal{M}, \mathbf{\Phi}, x$)
  return $\text{argmax}_{y\in\mathbf{GEN}(x)}\ \mathbf{S}(y|\mathcal{M}, \mathbf{\Phi}, x)$
end function

provided by the source classifier. Therefore, the performance of annotation transformation is correspondingly determined by the accuracy of the source classifier, and we can generate a more accurate parallel annotated corpus for better annotation adaptation if an improved source classifier can be obtained. Based on Model 2, two optimization strategies—iterative bidirectional training and predict-self hypothesis—are introduced to optimize the parallel annotated corpora for better annotation adaptation.

We first use an iterative training procedure to gradually improve the transformation accuracy by iteratively optimizing the parallel 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 parallel annotated corpora of the next iteration. Then in the new iteration, the better parallel annotated corpora will result in more accurate transfer classifiers, resulting in the generation of better transformed corpora.

Algorithm 4 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 on the target and source corpora, respectively (lines 2–5). In each training iteration, the transfer classifiers are trained on the current parallel annotated corpora (lines 7–8); they are used to produce the transformed corpora (lines 9–10), which provide better annotations for the parallel 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 (line 13).

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 transfer classifiers are better than the initial ones generated by the source and target classifiers, due to the assistance of the guiding features. In the following iterations,

Algorithm 4 Iterative annotation transformation.

1: **function** ITERANNOTRANS($\mathcal{C}_s$, $\mathcal{C}_t$)
2: $\mathcal{M}_s \leftarrow$ TRAIN($\mathcal{C}_s$) ▷ source classifier
3: $\mathcal{C}_t^s \leftarrow$ ANNOTATE($\mathcal{M}_s$, $\mathcal{C}_t$)
4: $\mathcal{M}_t \leftarrow$ TRAIN($\mathcal{C}_t$) ▷ target classifier
5: $\mathcal{C}_s^t \leftarrow$ ANNOTATE($\mathcal{M}_t$, $\mathcal{C}_s$)
6: **repeat**
7: $\mathcal{M}_{s\to t} \leftarrow$ TRANSTRAIN($\mathcal{C}_t^s$, $\mathcal{C}_t$) ▷ source-to-target transfer classifier
8: $\mathcal{M}_{t\to s} \leftarrow$ TRANSTRAIN($\mathcal{C}_s^t$, $\mathcal{C}_s$) ▷ target-to-source transfer classifier
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($\mathcal{C}_*^t$) ▷ enhanced classifier trained on merged corpus
13: **until** EVAL($\mathcal{M}_*$) converges
14: **return** $\mathcal{C}_*^t$
15: **end function**
16: **function** DECODE($\mathcal{M}$, $\mathbf{\Phi}$, x)
17: **return** $\text{argmax}_{y\in\mathbf{GEN}(x)}\ \mathbf{S}(y|\mathcal{M}, \mathbf{\Phi}, x)$
18: **end function**

the transformed corpora provide better annotations for the parallel annotated corpora of the subsequent iteration; the transformation accuracy will improve gradually along with the optimization of the parallel annotated corpora until convergence.

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

The most intuitive 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 procedure is performed on the source corpus to obtain a prediction that follows the target guideline; then a second, target-to-source transformation procedure is performed on this prediction result to check whether it can be transformed back to the original source annotation. The source-to-target prediction results that fail in this reverse-verification step are discarded, so this strategy can be called **predict-self filtering**.

A more sophisticated strategy can be called **predict-self re-estimation**. Instead of using the reversed transformation procedure for filtering, the re-estimation 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 is achieved. The scores of the two transformation models are weighted, integrated in a log-linear manner:

$$\mathbf{S}^{+}(y|\mathcal{M}_{s\to t}, \mathcal{M}_{t\to s}, \mathbf{\Phi}, x) = (1-\lambda) \times \mathbf{S}(y|\mathcal{M}_{s\to t}, \mathbf{\Phi}, x) + \lambda \times \mathbf{S}(x|\mathcal{M}_{t\to s}, \mathbf{\Phi}, y) \quad (3)$$

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

5. Experiments

To evaluate the performance of annotation adaptation, we experiment on two important NLP tasks, Chinese word segmentation and dependency parsing, both of which can be modeled as discriminative classification problems. For both tasks, we give the performances of the baseline models and the annotation adaptation algorithms.

5.1 Experimental Set-ups

We perform annotation adaptation for word segmentation from People's Daily (PD) (Yu et al. 2001) to Penn Chinese Treebank 5.0 (CTB) (Xue et al. 2005). The two corpora are built according to different segmentation guidelines and differ largely in quantity of data. CTB is smaller in size with about 0.5M words, whereas PD is much larger, containing nearly 6M words. Table 4 shows the data partitioning for the two corpora.

Table 4
Data partitioning for CTB and PD.

Partition	Sections	# of Words
CTB		
Training	$1-270$	0.47M
	$400-931$	
	$1,001-1,151$	
Developing	$301-325$	6.66K
Test	$271-300$	7.82K
PD		
Training	$02-06$	5.86M
Test	01	1.07M

To approximate more general scenarios of annotation adaptation problems, we extract from PD a subset that is comparable to CTB in size. Because there are many extremely long sentences in the original PD corpus, we first split them into normal sentences according to the full-stop punctuation symbol. We randomly select 20,000 sentences (0.45M words) from the PD training data as the new training set, and 1,000/1,000 sentences from the PD test data as the new testing/developing set. We label the smaller version of PD as SPD. The balanced source corpus and target corpus also facilitate the investigation of annotation adaptation.

Annotation adaptation for dependency parsing is performed from the CTB-derived syntactic dependency treebank (DCTB) (Yamada and Matsumoto 2003) to the Semantic Dependency Treebank (SDT) (Che et al. 2012). Semantic dependency encodes the semantic relationships between words, which are very different from syntactic dependencies. SDT is annotated on a small portion of the CTB text as depicted in Table 5; therefore, we use the subset of DCTB covering the remaining CTB text as the source corpus. We still denote the source corpus as DCTB in the following for simplicity.

5.2 Baseline Segmenters and Parsers

We train the baseline perceptron classifiers for Chinese word segmentation on the training sets of CTB and SPD, using corresponding development sets to determine the best

Table 5
Data partitioning for SDT.

Partition	Sections	# of Words
Training	$1-10$	244.44K
	$36-65$	
	$81-121$	
	$1,001-1,078$	
	$1,100-1,119$	
	$1,126-1,140$	
Developing	$66-80$	14.97K
	$1,120-1,125$	
Test	$11-35$	33.51K
	$1,141-1,151$	

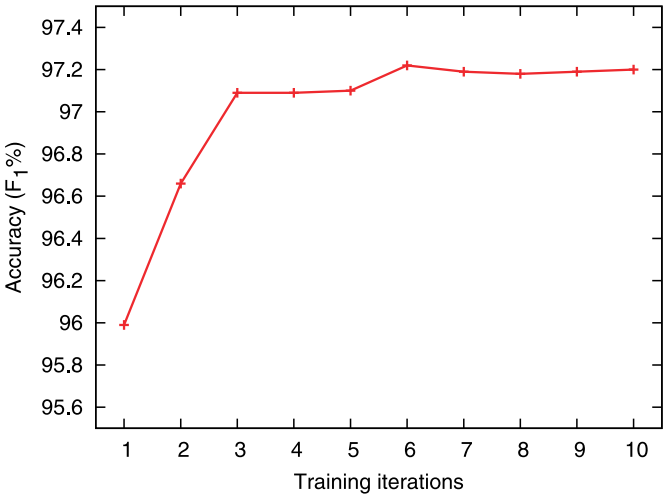


Figure 6
Learning curve of the averaged perceptron classifier on the CTB developing set.

training iterations. The performance measurement indicator for word segmentation is the balanced F-measure, $F = 2PR/(P + R)$, a function of Precision P and Recall R, where P is the percentage of words in segmentation results that are segmented correctly, and R is the percentage of correctly segmented words in the gold standard words.

For both syntactic and semantic dependency parsing, we concentrate on nonlabeled parsing that predicts the graphic dependency structure for the input sentence without considering dependency labels. The perceptron-based baseline dependency models are trained on the training sets of DCTB and SDT, using the development sets to determine the best training iterations. The performance measurement indicator for dependency parsing is the Unlabeled Attachment Score, denoted as Precision P, indicating the percentage of words in predicted dependency structure that are correctly attached to their head words.

Figure 6 shows the learning curve of the averaged perceptron for word segmentation on the development set. Accuracies of the baseline classifiers are listed in Table 6. We also report the performance of the classifiers on the testing sets of the opposite corpora. Experimental results are in line with our expectations. A classifier performs better in its corresponding testing set, and performs significantly worse on testing data following a different annotation guideline.

Table 7 shows the accuracies of the baseline syntactic and semantic parsers, as well as the performance of the parsers on the testing sets of the opposite corpora. Similar

Table 6
Performance of the baseline word segmenters.

	Test on (F_1%)	
Train on	CTB	SPD
CTB	97.35	86.65(↓ 10.70)
SPD	91.23(↓ 3.02)	94.25

Table 7
Performance of the baseline dependency parsers.

	Test on (P%)	
Train on	SDT	DCTB
SDT	77.51	53.62(↓ 23.89)
DCTB	51.92(↓ 32.62)	84.54

Table 8
Performance of automatic annotation adaptation models for word segmentation.

Model	**Time** (sec)	**Accuracy** (F_1%)
Merging	1.33	93.79
Model 1	4.39	97.67
Model 2	1.33	97.69
Model 3	1.33	**97.97**
Baseline	1.21	97.35

to the situations in word segmentation, two parsers give state-of-the-art accuracies on their own testing sets, but perform poorly on the other testing sets. This indicates the degree of divergence between the annotation guidelines of DCTB and SDT.

5.3 Automatic Annotation Adaptation

As a variant of Model 1, Model 2 shares the same transfer classifier, and differs only in training and decoding of the final classifier. Tables 8 and 9 show the performances of systems resulting from Models 1 and 2, as well as the classifiers trained on the directly merged corpora. The time costs for decoding are also listed to facilitate the practical comparison.

We find that the simple corpus merging strategy leads to a dramatic decrease in accuracy, due to the different and incompatible annotation guidelines. Model 1, the simplest model for annotation adaptation, gives significant improvement over the baseline classifiers for word segmentation and dependency parsing. This indicates that the statistical regularities for annotation adaptation learned by the transfer classifiers

Table 9
Performance of automatic annotation adaptation models for dependency parsing.

Model	**Time** (min)	**Accuracy** (P%)
Merging	7.38	65.95
Model 1	15.43	78.69
Model 2	7.38	78.73
Model 3	7.38	**79.34**
Baseline	6.67	77.51

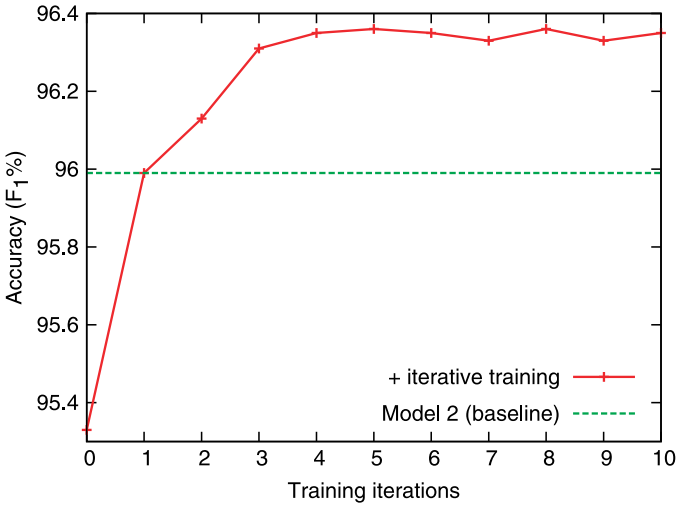


Figure 7
Training curve of the iterative training strategy over Model 2 for word segmentation.

bring performance improvement, utilizing guided decisions in the cascaded classifiers. Model 2 leads to classifiers with accuracy increments comparable to those of Model 1, while consuming only one third of the decoding time. It is inconsistent with our expectation. The strategy of directly transforming the source corpus to the target guideline also facilitates the utilization of more than one source corpus.

We first introduce the iterative training strategy to Model 2. The corresponding development sets are used to determine the best training iterations for the iterative annotation transformations. After each iteration, we test the performance of the classifiers trained on the merged corpora. Figures 7 and 8 show the performance curves for Chinese word segmentation and semantic dependency parsing, respectively, with iterations ranging from 1 to 10. The performance of Model 2 is naturally included

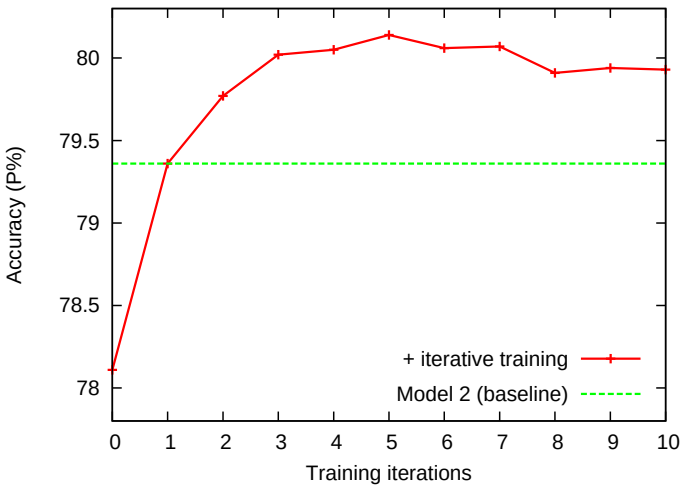


Figure 8
Training curve of the iterative training strategy over Model 2 for dependency parsing.

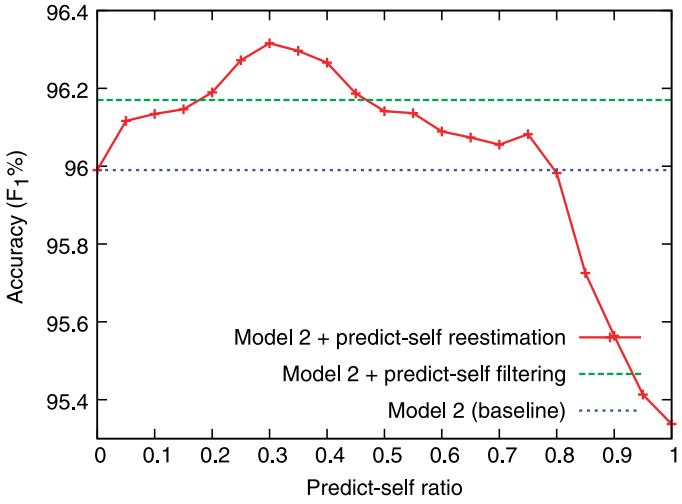


Figure 9
Performance of predict-self filtering and predict-self re-estimation over Model 2 for word segmentation.

in the curves (located at iteration 1). The curves show that, for both segmentation and parsing, the accuracies of the classifiers trained on the merged corpora consistently improve in the earlier iterations (e.g., from iteration 2 to iteration 5 for word segmentation).

Experiments for introduction of predict-self filtering and predict-self re-estimation are shown in Figures 9 and 10. The curves show the performances of the predict-self re-estimation with a series of weight parameters, ranging from 0 to 1 with step 0.05. Note that in both figures, the points at $\lambda = 0$ show the performances of Model 2. We find that predict-self filtering brings a slight improvement over the baseline for

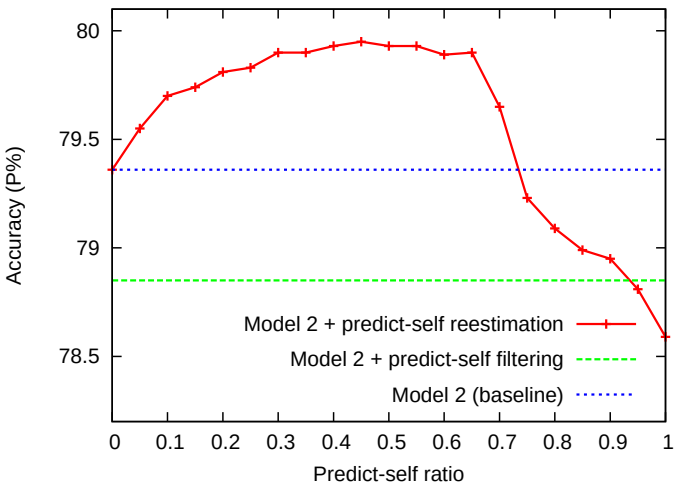


Figure 10
Performance of predict-self filtering and predict-self re-estimation over Model 2 for dependency parsing.

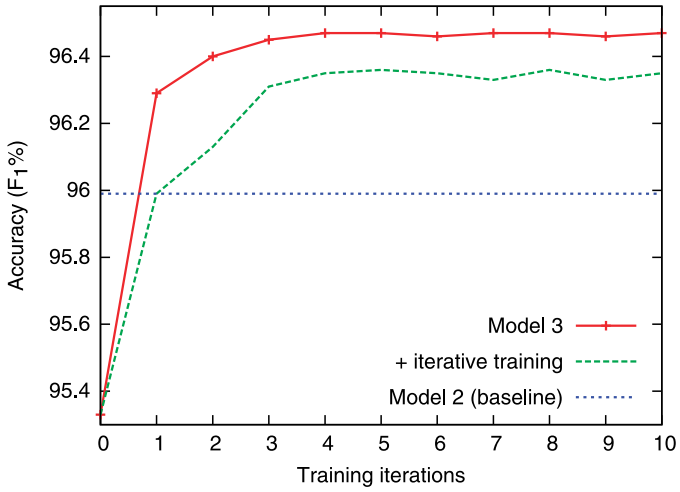


Figure 11
Training curve of annotation adaptation with Model 3 for word segmentation.

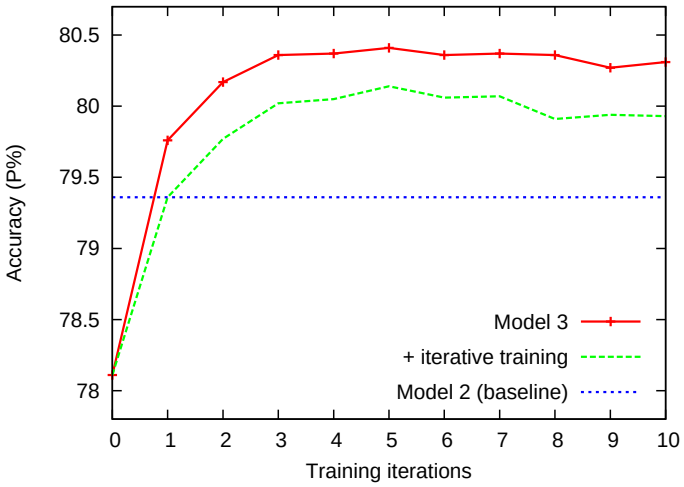


Figure 12
Training curve of annotation adaptation with Model 3 for dependency parsing.

word segmentation, but even decreases the accuracy for dependency parsing. An initial analysis on the experimental results reveals that the filtering strategy discards some complicated sentences in the source corpora, and the discarded sentences would bring further improvement if properly used. For example, in word segmentation, predict-self filtering discards 5% of sentences from the source corpus, containing nearly 10% of the training words. For the two tasks, the predict-self re-estimation outperforms the filtering strategy. With properly tuned weights, predict-self re-estimation can make better use of the training data. The largest accuracy improvements achieved over Model 2 for word segmentation and dependency parsing are 0.3 points and 0.6 points.

Figures 11 and 12 show the performance curves after the introduction of both iterative training and predict-self re-estimation on the basis of Model 2 (this enhanced

Table 10
The performance of the enhanced annotation adaptation for word segmentation, compared with previous work.

Model	**Accuracy** (F_1%)
SPD → CTB	97.97
PD → CTB	**98.43**
Previous Work	
(Jiang et al. 2008)	97.85
(Kruengkrai et al. 2009)	97.87
(Zhang and Clark 2010)	97.79
(Sun 2011)	98.17

Table 11
The performance of the enhanced annotation adaptation for dependency parsing, compared with the results of SemEval-2012 contest, where the best systems of each institute are listed.

Model	**Accuracy** (P%)
DCTB → SDT	79.34
SemEval-2012 Contest	
Zhijun Wu-1	80.64
Zhou Qiaoli-3	80.60
NJU-Parser-1	80.35
ICT-1	73.20
Giuseppe Attardi-SVM-1-R	60.83

model is denoted as Model 3). We find that the predict-self re-estimation brings improvement to the iterative training at each iteration, for both word segmentation and dependency parsing. The maximum performance is achieved at iteration 4 for word segmentation, and at iteration 5 for dependency parsing. The corresponding models are evaluated on the corresponding testing sets, and the experimental results are also shown in Tables 8 and 9. Compared to Model 1, the optimized annotation adaptation strategy, Model 3, leads to classifiers with significantly higher accuracy and to processing speeds that are several times faster. Tables 10 and 11 show the experimental results compared with previous work. For both Chinese word segmentation and semantic dependency parsing, automatic annotation adaptation yields state-of-the-art performance, despite using single classifiers with only local features. Note that for the systems in the SemEval contest (Che et al. 2012), many other technologies including clause segmentation, system combination, and complicated features were adopted, as well as elaborate engineering. We also performed significance tests[2] to verify the effectiveness of annotation adaptation.We find that for both Chinese word segmentation and semantic dependency parsing, annotation adaptation brings significant improvement ($p < 0.001$) over the baselines trained on the target corpora only.

2 `http://www.cis.upenn.edu/~dbikel/download/compare.pl`.

Table 12
Quantitative analysis for performance of annotation adaptation on word segmentation. The words are grouped according to POS tags, and for each group, the recall values of baseline and annotation adaptation are reported. To save space, only the categories with proportions of more than 1% and with performance fluctuations of more than 0.1 points are listed.

Word Type	Proportion	Baseline	Anno. Ada.	Trend
AD	3.99	98.75	96.87	↓
CD	2.07	98.79	99.39	↑
JJ	3.25	95.01	95.40	↑
LC	1.36	98.16	99.08	↑
NN	29.95	97.79	98.83	↑
NR	9.81	97.32	98.98	↑
VA	1.58	95.27	97.63	↑
VV	13.52	98.33	98.61	↑

To evaluate the stability of annotation adaptation, we perform quantitative analysis on the results of annotation adaptation. For word segmentation, the words are grouped according to POS tags. For dependency parsing, the dependency edges are grouped according to POS tag pairs. For each category, the recall values of baseline and annotation adaptation are reported. To filter the lists, we set two significance thresholds with respect to the proportion of a category and the performance fluctuation between two systems. For word segmentation, only the categories with proportions of more than 1% and with fluctuations of more than 0.1 points are reserved, and for dependency parsing, the two thresholds are 1% and 0.5. Tables 12 and 13 show the analysis results for word segmentation and dependency parsing, respectively. For both tasks, annotation adaptation brings improvement for most of the situations.

Table 13
Quantitative analysis for performance of annotation adaptation on dependency parsing. The dependency edges are grouped according to POS tag pairs, and for each group, the recall values of baseline and annotation adaptation are reported. To save space, only the categories with proportions of more than 1% and with performance fluctuations of more than 0.5 points are listed.

Edge Type	Proportion	Baseline	Anno. Ada.	Trend
NN→JJ	2.61	90.76	93.41	↑
NN→LC	1.00	98.19	97.29	↓
NN→M	1.68	83.48	87.43	↑
NN→NR	3.14	79.82	83.86	↑
NN→P	2.31	85.49	84.83	↓
NN→PU	3.15	75.71	76.39	↑
NN→VV	1.89	64.64	71.81	↑
VC→PU	1.33	67.87	70.13	↑
VE→NN	1.07	77.80	82.86	↑
VV→AD	5.55	81.69	83.00	↑
VV→DEC	1.76	93.82	94.51	↑
VV→NN	11.36	78.85	82.02	↑
VV→NR	1.71	76.19	79.01	↑
VV→PU	9.89	64.40	66.91	↑
VV→VV	7.72	59.74	62.21	↑

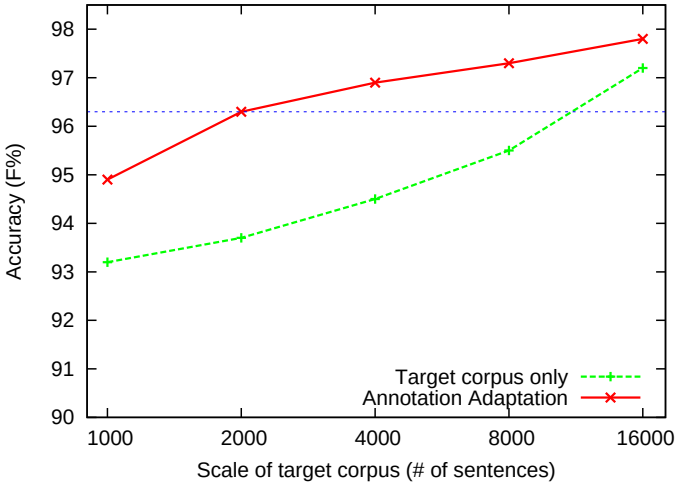


Figure 13
Performance of annotation adaptation with varying-size target corpora for Chinese word segmentation.

We further investigate the effect of varying the sizes of the target corpora. Experiments are conducted for word segmentation and dependency parsing with fixed-size source corpora and varying-size target corpora. We use SPD and DCTB as the source corpora for word segmentation and dependency parsing, respectively. Figures 13 and 14 show the performance curves on the testing sets. We find that, for both word segmentation and dependency parsing, the improvements brought by annotation adaptation are more significant when the target corpora are smaller. It means that the automatic annotation adaptation is more valuable when the size of the target corpus is small,

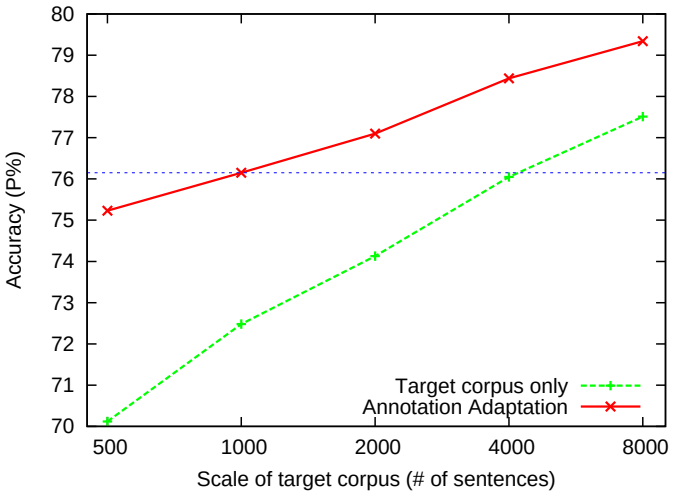


Figure 14
Performance of annotation adaptation with varying-size target corpora for semantic dependency parsing.

which is good news for the situation where the corpus we are concerned with is smaller but a larger differently annotated corpus exists.

Of course, the comparison between automatic annotation adaptation and previous strategies without using additional training data is unfair. Our work aims to find another way to improve NLP tasks: focusing on the collection of more training data instead of making full use of a certain corpus. We believe that the performance of automatic annotation adaptation can be further improved by adopting the advanced technologies of previous work, such as complicated features and model combination. It would be useful to conduct experiments with more source-annotated training data, such as the SIGHAN data set for word segmentation, to investigate the trend of improvement along with the further increment of annotated sentences. It would also be valuable to evaluate the improved word segmenter and dependency parser on the out-of-domain data sets. However, currently most corpora for word segmentation and dependency parsing do not explicitly distinguish the domains of their data sections, making such evaluations difficult to conduct.

6. Discussion: Application Situations

Automatic annotation adaptation aims to transform the annotations in a corpus to the annotations following other guidelines. The models for annotation adaptation use a transfer classifier to learn the statistical correspondence regularities between different annotation guidelines. These statistical regularities are learned from a parallel annotated corpus, which does not need to be manually annotated. In fact, the models for annotation adaptation train the transfer classifier on an *automatically generated* parallel annotated corpus, which is generated by processing a corpus with a classifier trained on another corpus. That is to say, if we want to conduct annotation adaptation across several corpora, no additional corpora need to be manually annotated. This setting makes the strategy of annotation adaptation more general, because it is much harder to manually annotate a parallel annotated corpus, regardless of the language or the NLP problem under consideration. To tackle the problem of noise in automatically generated annotations, the advanced models we designed generate a better parallel annotated corpus by making use of strategies such as iterative optimization.

Automatic annotation adaptation can be applied in any situation where we have multiple corpora with different and incompatible annotation philosophies for the same task. As our case studies, both Chinese word segmentation and dependency parsing have more than one corpora with different annotation guidelines, such as the People's Daily and the Penn Chinese Treebank for Chinese word segmentation. In a more abstract view, constituency grammar and dependency grammar can be treated as two annotation guidelines for parsing. The syntactic knowledge in a constituency treebank and a dependency treebank, therefore, can be integrated by automatic annotation adaptation. For example, the LinGo Redwoods Treebank can also be transformed to the annotation guideline of the Semantic Dependency Treebank.

Furthermore, the annotations (such as a grammar) given by bilingual projection or unsupervised induction can be seen as following a special annotation philosophy. For bilingually projected annotations, the annotation guideline would be similar to that of the counterpart language. For unsupervised induced annotations, the annotation guideline reflects the statistical structural distribution of a specific data set. In both situations, the underlying annotation guidelines may be largely different from that of the testing sets, which usually come from human-annotated corpora. The system trained on a bilingually projected or unsupervised induced corpus may perform poorly on an existing

testing set, but if the projected or induced corpus has high inner consistency, it could improve a system trained on an existing corpus by automatic annotation adaptation. In this point of view, the practical value of the current work on bilingual projection and unsupervised induction may be underestimated, and annotation adaptation could make better use of the projected or induced knowledge.[3]

7. Related Work

There has already been some preliminary work tackling the divergence between different annotation guidelines. Gao et al. (2004) described a transformation-based converter to transfer a certain word segmentation result to another annotation guideline. They designed class-type transformation templates and used the transformation-based error-driven learning method of Brill (1995) to learn what word delimiters should be modified. Many efforts have been devoted to manual treebank transformation, where PTB is adapted to other grammar formalisms, such as CCG and LFG (Cahill et al. 2002; Hockenmaier and Steedman 2007). However, all these are heuristic-based—that is, they need manually designed transformation templates and involve heavy human engineering. Such strategies are hard to be generalized to POS tagging, not to mention other complicated structural prediction tasks.

We investigated the automatic integration of word segmentation knowledge in differently annotated corpora (Jiang, Huang, and Liu 2009; Jiang et al. 2012), which can be seen as the preliminary work of automatic annotation adaptation. Motivated by our initial investigation, researchers applied similar methodologies to constituency parsing (Sun, Wang, and Zhang 2010; Zhu, Zhu, and Hu 2011) and word segmentation (Sun and Wan 2012). This previous work verified the effectiveness of automatic annotation adaptation, but did not reveal the essential definition of the problem nor the intrinsic principles of the solutions. Instead, this work clearly defines the problem of annotation adaptation, reveals the intrinsic principles of the solutions, and systematically describes a series of gradually improved models. The most advanced model learns transformation regularities much better and achieves significant higher accuracy for both word segmentation and dependency parsing, without slowing down the final language processors.

The problem of automatic annotation adaptation can be seen as a special case of transfer learning (Pan and Yang 2010), where the source and target tasks are similar, but not identical. More specifically, the problem related to annotation adaptation assumes that the labeling mechanism across the source and target tasks are the same, but the predictive functions are different. The goal of annotation adaptation is to adapt the source predictive function to be used for the target task by exploiting the labeled data of the source task and the target task. Furthermore, automatic annotation adaptation approximately falls into the spectrum of relational-knowledge-transfer problems (Mihalkova, Huynh, and Mooney 2007; Mihalkova and Mooney 2008; Davis and Domingos 2009), but it tackles problems where the relations among data between the source and target domains can be largely isomerous—or, in other words, with different and incompatible annotation schemes. This work enriches the research of transfer learning by proposing and solving an NLP problem different from previous situations. For more details of transfer learning please refer to the survey of Pan and Yang (2010).

3 We have performed preliminary experiments on word segmentation. Bilingual projection was conducted from English to Chinese with the Chinese–English FBIS as the bilingual corpus. By annotation adaptation, the projected corpus for word segmentation brings a significant F-measure increment of nearly 0.6 points over the baseline trained on CTB only.

The training procedure for an annotation adaptation model requires a parallel annotated corpus (which may be automatically generated); this fact puts the method into the neighborhood of the family of approaches known as annotation projection (Hwa et al. 2002, 2005; Ganchev, Gillenwater, and Taskar 2009; Smith and Eisner 2009; Jiang and Liu 2010; Das and Petrov 2011). Essentially, annotation adaptation and annotation projection tackle different problems; the former aims to transform the annotations from one *guideline* to another (of course in the same language), whereas the latter aims to project the annotation (as well as the annotation guideline) from one *language* to another. Therefore, the machine learning methods for annotation adaptation pay attention to automatic transformation of annotations, while for annotation projection, the machine learning methods focus on the bilingual projection across languages.

Co-training (Sarkar 2001) and classifier combination (Nivre and McDonald 2008) are two techniques for training improved dependency parsers. The co-training technique lets two different parsing models learn from each other during the parsing of unlabeled text: One model selects some unlabeled sentences it can confidently parse, and provides them to the other model as additional training data in order to train more powerful parsers. The classifier combination lets graph-based and transition-based dependency parsers utilize the features extracted from each other's parsing results, to obtain combined, enhanced parsers. The two techniques aim to let two models learn from each other on the same corpus with the same distribution and annotation guideline, whereas our strategy aims to integrate the knowledge in multiple corpora with different annotation guidelines.

The iterative training procedure used in the optimized model shares some similarities with the co-training algorithm in parsing (Sarkar 2001), where the training procedure lets two different models learn from each other during parsing of the raw text. The key idea of co-training is to utilize the complementarity of different parsing models to mine additional training data from raw text, whereas iterative training for annotation adaptation emphasizes the iterative optimization of the parallel annotated corpora used to train the transfer classifiers. The predict-self methodology is implicit in many unsupervised learning approaches; it has been successfully used in unsupervised dependency parsing (Daumé III 2009). We adapt this idea to the scenario of annotation adaptation to improve transformation accuracy.

In recent years much effort has been devoted to the improvement of word segmentation and dependency parsing. For example, the introduction of global training or complicated features (Zhang and Clark 2007, 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, Zong, and Su 2010; Sun 2011); and the semi-supervised and unsupervised technologies utilizing raw text (Zhao and Kit 2008; Johnson and Goldwater 2009; Mochihashi, Yamada, and Ueda 2009; Hewlett and Cohen 2011). We believe that the annotation adaptation technologies can be adopted jointly with complicated features, system combination, and semi-supervised/unsupervised technologies to further improve the performance of word segmentation and dependency parsing.

8. Conclusion and Future Work

We have described the problem of annotation adaptation and the intrinsic principles of its solutions, and proposed a series of successively enhanced models that can automatically adapt the divergence between different annotation formats. These models learn the statistical regularities of adaptation between different annotation guidelines,

and integrate the knowledge in corpora with different annotation guidelines. In the problems of Chinese word segmentation and semantic dependency parsing, annotation adaptation algorithms bring significant improvements by integrating the knowledge in differently annotated corpora, People's Daily and Penn Chinese Treebank for word segmentation, and Penn Chinese Treebank and Semantic Dependency Parsing for dependency parsing. For both tasks, annotation adaptation leads to a segmenter and a parser achieving the state-of-the-art, despite using only local features in single classifiers.

Many aspects related to annotation adaptation deserve further investigation in the future. First, models for annotation adaptation can be adapted to other NLP tasks such as semantic analysis. Second, jointly tackling the divergences in both annotations and domains is an important problem. In addition, an unsupervised-induced or bilingually projected corpus, despite performing poorly on the specified testing data, may have high inner annotation consistency. That is to say, the induced corpora can be treated as a knowledge source following another annotation guideline, and the performance of current unsupervised or bilingually projected models may be seriously underestimated. Annotation adaptation may give us a new perspective on knowledge induction and measurement for such methods.

Acknowledgments

Jiang, Lü, and Liu were supported by National Natural Science Foundation of China (contract 61202216) and the National Key Technology R&D Program (no. 2012BAH39B03). Huang was supported in part by the DARPA DEFT Project (FA8750-13-2-0041). Liu was partially supported by the Science Foundation Ireland (grant no. 07/CE/I1142) as part of the CNGL at Dublin City University. We also thank the anonymous reviewers for their insightful comments. Finally, we want to thank Chris Hokamp for proofreading.

References

Blitzer, John, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of EMNLP*, pages 120–128, Sydney.

Brill, Eric. 1995. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565.

Buchholz, Sabine and Erwin Marsi. 2006. CONLL-X shared task on multilingual dependency parsing. In *Proceedings of CoNLL*, pages 149–164, New York, NY.

Cahill, Aoife, Mairead McCarthy, Josef van Genabith, and Andy Way. 2002. Automatic annotation of the Penn treebank with LFG F-structure information. In *Proceedings of the LREC Workshop*, Las Palmas.

Che, Wanxiang, Meishan Zhang, Yanqiu Shao, and Ting Liu. 2012. Semeval-2012 task 5: Chinese semantic dependency parsing. In *Proceedings of SemEval*, pages 378–384, Montreal.

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

Das, Dipanjan and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of ACL*, pages 600–609, Portland, OR.

Daumé III, Hal. 2007. Frustratingly easy domain adaptation. In *Proceedings of ACL*, pages 256–263, Prague.

Daumé III, Hal. 2009. Unsupervised search-based structured prediction. In *Proceedings of ICML*, pages 209–216, Montreal.

Daumé III, Hal and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26:101–126.

Davis, Jesse and Pedro Domingos. 2009. Deep transfer via second-order Markov logic. In *Proceedings of ICML*, pages 217–224, Montreal.

Eisner, Jason M. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of COLING*, pages 340–345, Copenhagen.

Ganchev, Kuzman, Jennifer Gillenwater, and Ben Taskar. 2009. Dependency grammar induction via bitext projection constraints. In *Proceedings of ACL*, pages 369–377, Singapore.

Gao, Jianfeng, Andi Wu, Mu Li, Chang-Ning Huang, Hongqiao Li, Xinsong Xia, and Haowei Qin. 2004. Adaptive Chinese word segmentation. In *Proceedings of ACL*, pages 462–469, Barcelona.

Hewlett, Daniel and Paul Cohen. 2011. Fully unsupervised word segmentation with BVE and MDL. In *Proceedings of ACL*, pages 540–545, Portland, OR.

Hockenmaier, Julia and Mark Steedman. 2007. CCGBank: A corpus of CCG derivations and dependency structures extracted from the Penn treebank. *Computational Linguistics*, 33(3):355–396.

Hwa, Rebecca, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural Language Engineering*, 11(3):311–325.

Hwa, Rebecca, Philip Resnik, Amy Weinberg, and Okan Kolak. 2002. Evaluating translational correspondence using annotation projection. In *Proceedings of ACL*, pages 392–399, Philadephia, PA.

Jiang, Wenbin, Liang Huang, and Qun Liu. 2009. Automatic adaptation of annotation standards: Chinese word segmentation and POS tagging – A case study. In *Proceedings of ACL*, pages 522–530, Singapore.

Jiang, Wenbin, Liang Huang, Yajuan Lü, and Qun Liu. 2008. A cascaded linear model for joint Chinese word segmentation and part-of-speech tagging. In *Proceedings of ACL*, pages 897–904, Columbus, OH.

Jiang, Wenbin and Qun Liu. 2010. Dependency parsing and projection based on word-pair classification. In *Proceedings of the ACL*, pages 12–20, Uppsala.

Jiang, Wenbin, Fandong Meng, Qun Liu, and Yajuan Lü. 2012. Iterative annotation transformation with predict-self reestimation for Chinese word segmentation. In *Proceedings of EMNLP*, pages 412–420, Jeju Island.

Johnson, Mark and Sharon Goldwater. 2009. Improving nonparameteric Bayesian inference: Experiments on unsupervised word segmentation with adaptor grammars. In *Proceedings of NAACL*, pages 317–325, Boulder, CO.

Kruengkrai, Canasai, 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*, pages 513–521, Singapore.

Li, Zhongguo. 2011. Parsing the internal structure of words: A new paradigm for Chinese word segmentation. In *Proceedings of ACL*, pages 1,405–1,414, Portland, OR.

Marcus, Mitchell P., Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2):313–330.

Martins, André F. T., Dipanjan Das, Noah A. Smith, and Eric P. Xing. 2008. Stacking dependency parsers. In *Proceedings of EMNLP*, pages 157–166, Honolulu, HI.

McDonald, Ryan, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL*, pages 91–98, Ann Arbor, MI.

McDonald, Ryan and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL*, pages 81–88, Trento.

Mihalkova, Lilyana, Tuyen Huynh, and Raymond J. Mooney. 2007. Mapping and revising Markov logic networks for transfer learning. In *Proceedings of AAAI*, volume 7, pages 608–614, Vancouver.

Mihalkova, Lilyana and Raymond J. Mooney. 2008. Transfer learning by mapping with minimal target data. In *Proceedings of AAAI Workshop Transfer Learning for Complex Tasks*, Chicago, IL.

Mochihashi, Daichi, Takeshi Yamada, and Naonori Ueda. 2009. Bayesian unsupervised word segmentation with nested Pitman-Yor language modeling. In *Proceedings of ACL-IJCNLP*, pages 100–108, Singapore.

Nakagawa, Tetsuji and Kiyotaka Uchimoto. 2007. A hybrid approach to word segmentation and POS tagging. In *Proceedings of ACL*, pages 217–220, Prague.

Ng, Hwee Tou 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*, pages 277–284, Barcelona.

Nivre, Joakim and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL*, pages 950–958, Columbus, OH.

Oepen, Stephan, Kristina Toutanova, Stuart Shieber, Christopher Manning Dan Flickinger, and Thorsten Brants. 2002. The LinGo Redwoods treebank: Motivation and preliminary applications. In *Proceedings of COLING*, volume 2, pages 1–5, Taipei.

Pan, Sinno Jialin and Qiang Yang. 2010. A survey on transfer learning. *IEEE TKDE*, 22(10):1345–1359.
Sarkar, Anoop. 2001. Applying co-training methods to statistical parsing. In *Proceedings of NAACL*, pages 1–8, Pittsburgh, PA.
Smith, David and Jason Eisner. 2009. Parser adaptation and projection with quasi-synchronous grammar features. In *Proceedings of EMNLP*, volume 2, pages 822–831, Singapore.
Sun, Weiwei. 2011. A stacked sub-word model for joint Chinese word segmentation and part-of-speech tagging. In *Proceedings of ACL*, pages 1,385–1,394, Portland, OR.
Sun, Weiwei and Xiaojun Wan. 2012. Reducing approximation and estimation errors for Chinese lexical processing with heterogeneous annotations. In *Proceedings of ACL*, volume 1, pages 232–241, Jeju Island.
Sun, Weiwei, Rui Wang, and Yi Zhang. 2010. Discriminative parse reranking for Chinese with homogeneous and heterogeneous annotations. In *Proceedings of CIPS-SIGHAN*, Beijing. Available at `http://aclweb.org/anthology/W10-4144`.
Wang, Kun, Chengqing Zong, and Keh-Yih Su. 2010. A character-based joint model for Chinese word segmentation. In *Proceedings of COLING*, pages 1,173–1,181, Beijing.
Xue, Nianwen and Libin Shen. 2003. Chinese word segmentation as LMR tagging. In *Proceedings of SIGHAN Workshop*, volume 17, pages 176–179, Sapporo.
Xue, Nianwen, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The Penn Chinese treebank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11(2):207–238.
Yamada, H. and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*, pages 195–206, Nancy.
Yu, Shiwen, 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, Institute of Computational Linguistics, Peking University.
Zhang, Yue and Stephen Clark. 2007. Chinese segmentation with a word-based perceptron algorithm. In *Proceedings of ACL*, pages 840–847, Prague.
Zhang, Yue and Stephen Clark. 2010. A fast decoder for joint word segmentation and POS-tagging using a single discriminative model. In *Proceedings of EMNLP*, pages 843–852, Cambridge, MA.
Zhao, Hai and Chunyu Kit. 2008. Unsupervised segmentation helps supervised learning of character tagging for word segmentation and named entity recognition. In *Proceedings of IJCNLP*, pages 106–111, Hyderabad.
Zhu, Muhua, Jingbo Zhu, and Minghan Hu. 2011. Better automatic treebank conversion using a feature-based approach. In *Proceedings of ACL*, volume 2, pages 715–719, Portland, OR.