Automatic Treebank Conversion via Informed Decoding - A Case Study on Chinese Treebanks

MUHUA ZHU, JINGBO ZHU, and TONG XIAO, Northeastern University, China

Treebanks are valuable resources for syntactic parsing. For some languages such as Chinese, we can obtain multiple constituency treebanks which are developed by different organizations. However, due to discrepancies of underlying annotation standards, such treebanks in general cannot be used together through direct data combination. To enlarge training data for syntactic parsing, we focus in this article on the challenge of unifying standards of disparate treebanks by automatically converting one treebank (source treebank) to fit a different standard which is exhibited by another treebank (target treebank).

We propose to convert a treebank in two sequential steps which correspond to the part-of-speech level and syntactic structure level (including tree structures and grammar labels), respectively. Approaches used in both levels can be unified as an *informed decoding* procedure, where information derived from original annotation in a source treebank is used to guide the conversion conducted by a POS tagger (or a parser in the syntactic structure level) trained on a target treebank. We take two Chinese treebanks as a case study, and experiments on these two treebanks show significant improvements in conversion accuracy over baseline systems, especially in situations where a target treebank is small in size.

Categories and Subject Descriptors: I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Language parsing and understanding*

General Terms: Documentation, Languages

Additional Key Words and Phrases: Chinese POS tagging, Chinese syntactic parsing, informed decoding, treebank conversion

ACM Reference Format:
Zhu, M., Zhu, J., and Xiao, T. 2011. Automatic treebank conversion via informed decoding - A case study on Chinese treebanks. ACM Trans. Asian Lang. Inform. Process. 10, 3, Article 12 (September 2011), 24 pages. DOI = 10.1145/2002980.2002982 http://doi.acm.org/10.1145/2002980.2002982

1. INTRODUCTION

Machine learning methods have been extensively applied to natural language processing problems in recent years. Typically, increase in the scale of training data boosts the performance of machine learning methods, which in turn enhances the quality of learning-based NLP systems [Banko and Brill 2001]. However, annotating data by humans is time consuming and labor intensive. For this reason, manually annotated corpora are considered as the most valuable resources for NLP.

This work was supported in part by the National Science Foundation of China (60873091; 61073140), Specialized Research Fund for the Doctoral Program of Higher Education (20100042110031), the Fundamental Research Funds for the Central Universities.
Authors' address: M. Zhu; email: zhumuhua@gmail.com; J. Zhu (corresponding author); email: zhujingbo@mail.neu.edu.cn; T. Xiao; email: xiaotong@mail.neu.edu.cn, the Key Laboratory of Medical Image Computing (Ministry of Education), the Natural Language Processing Lab, Northeastern University, Shenyang 110819, China.

DOI 10.1145/2002980.2002982 http://doi.acm.org/10.1145/2002980.2002982

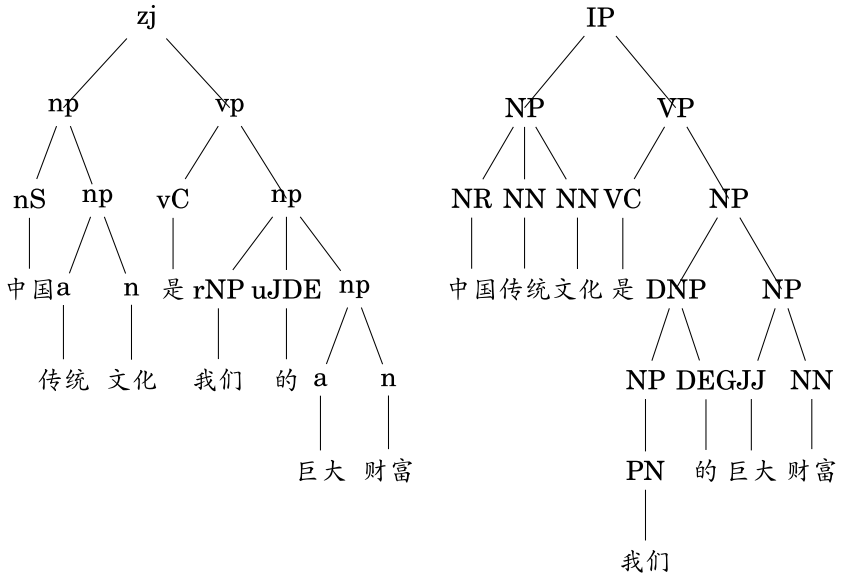

Fig. 1. Example trees with TCT (left) and CTB (right) annotations.

In practice, there often exist more than one corpus for the same NLP tasks. For example, for constituent (phrase-structure) syntactic parsing [Charniak 2000; Collins 1999; Petrov et al. 2006] of Chinese, besides the most popular treebank, Penn Chinese Treebank (CTB) [Xue et al. 2002], there are other treebanks such as Tsinghua Chinese Treebank (TCT) [Zhou 1996] and Peking Chinese Treebank.[1] In order to make full use of manually constructed resources, it is practically significant to use such corpora jointly. At first sight, direct combination of multiple corpora is a way to this end. However, corpora built by different organizations often follow different annotation standards and/or even different linguistic theories. We take treebanks CTB and TCT as a case study (see Figure 1). Although both CTB and TCT are Chomskian-style treebanks, they have annotation divergences in at least two dimensions: a) CTB and TCT use dramatically different label sets for both parts-of-speech and grammar categories and b) CTB and TCT have distinct hierarchical structures. For example, the Chinese words "中国 (Chinese) 传统(traditional) 文化 (culture)" are labeled as a flat noun phrase according to the CTB standard (right side in Figure 1), but in TCT, the last two words are instead grouped together beforehand (left side in Figure 1). Since almost all state-of-the-art syntactic parsers assume homogeneous annotation in the training corpus, heterogenous treebanks, such as CTB and TCT, are often used independently.

To enlarge labeled training data for constituent syntactic parsing, we focus in this article on unifying annotations in heterogeneous treebanks, say CTB and TCT, through treebank conversion. The task of treebank conversion is defined to convert annotation

[1]The treebank is not publicly available when this article is written; acquire a brief description (in Chinese) of this treebank by following the URL. See http://ccl.pku.edu.cn:8080/WebTreebank/WebTreebank_Readme.html.

in one treebank (source treebank) to fit a desired annotation standard which is exhibited by another treebank (target treebank). To this end, we view a parse tree of a sentence as consisting of two syntactic levels: part-of-speech and syntactic structure (including tree structures and grammar labels). Treebank conversion proceeds correspondingly in these two levels. Our approaches build on the idea of *informed decoding* where a POS tagger (a parser in the syntactic structure level) trained on a target treebank automatically assigns POS tags (parse trees for syntactic structure conversion) to the sentences in a source treebank with original annotation in the source treebank being used as guiding information. The proposed approaches are language independent, though, we take Chinese treebanks as a case study and evaluate accuracy of the approaches on two open Chinese treebanks: CTB and TCT. Experimental results show that our approaches achieve significant improvements over baseline systems, especially when training data used for building the POS tagger (parser) is limited in size. The reason why improvements on a small-scale target treebank are larger than those achieved on a large-scale target treebank is two-fold. First, we consider the relation between performance of baseline systems and the size of a target treebank. Specifically, performance of baseline systems is dependent on the accuracy of POS taggers (or syntactic parsers) built on a target treebank, so the size of a target treebank affects the performance of POS taggers (or syntactic parsers) which in turn affects the performance of baseline systems. Second, performance of our approach, informed decoding, is also dependent on the accuracy of POS taggers (or syntactic parsers). However, in situations where we have only a small target treebank, information (quite reliable in some sense) derived from a source treebank can play a more important role. Actually, such a result is quite satisfactory because it is the situations where we do not have enough target data that drive us to enlarge target data through treebank conversion.

This article is an extension of our previous work [Zhu and Zhu 2010; Zhu et al. 2009]. By contrast, this article adds one section (Section 2.2) dedicated to comparisons of the heterogeneous treebanks used in this article. Such comparisons are informative to the design of treebank conversion approaches. Moreover, we reexamine the approach to POS tag conversion and unify POS tag conversion and syntactic structure conversion in a framework termed *informed decoding*: both of them are based on the idea of incorporating information derived from a source dataset into the decoding phase of a language analyzer (a POS tagger or a syntactic parser). Such unification is a contribution in some sense, since viewing annotation conversion from the perspective of informed decoding inspires us to use the same idea to solve all the annotation conversion problems. In other words, it is always possible to conduct annotation conversion by extending decoders of readily available analyzers, such POS tagger and syntactic parser.

The rest of this article is structured as follows. Section 2 introduces the treebanks used for experiments and also presents a comparison of the treebanks. In Sections 3 and 4, approaches and experimental results are presented for POS and syntactic structure conversion respectively. In Section 5 we briefly review previous work on treebank conversion. Finally, Section 6 concludes this work.

2. DATA PREPARATION AND ANALYSIS

2.1 Experimental Data

In the experiments of this article, we always use Penn Chinese Treebank (CTB) as the target treebank and Tsinghua Chinese Treebank (TCT) as the source treebank. That is, the CTB standard is the one we are interested in. With regard to the CTB corpus, we split CTB 5.1 following the conventional scheme: articles 001-270 and 400-1151 are used for training, articles 271-300 are used as test data, and articles 301-325 are used

Table I. Basic Statistics on the CTB and TCT Data

CTB-Partitions	Train	Dev	Test
#Sentences	18,100	352	348
#Words	493,869	6,821	8,008
Ave-Length	27.29	19.38	23.01
TCT-Partitions	Train	Dev	Test
#Sentences	17,529	N/A	1,000
#Words	481,061	N/A	26,381
Ave-Length	27.44	N/A	26.38

as development data.[2] Concerning the TCT corpus, we use the data released by CIPS-SIGHAN-2010 syntactic parsing competition,[3] which is a portion of the whole TCT corpus. In that competition, the dataset is divided into two parts: 17,529 sentences for training and 1,000 sentences for testing. In order to keep the notation uncluttered, we still use *TCT* to refer to this released dataset. Table I contains statistics on CTB and TCT.

In this article, we ran two groups of experiments: a) we evaluate annotation conversion accuracy in the POS and syntactic structure levels, respectively and b) after converting the TCT training data, we go one step further to measure the efficacy of newly gained data on boosting the performance of POS taggers and syntactic parsers trained on CTB. For the first group of experiments, we had three annotators manually assign TCT-style parse trees to the sentences extracted from the CTB test set. Thus, there are two parse trees, CTB-style and TCT-style, available for each sentence in the CTB test set. For convenience of reference, this new test set is referred to as *CTB-Conversion-Set*, which is used for evaluation of accuracy of both POS and syntactic structure conversion. For POS conversion evaluation, we additionally sampled 1,000 sentences from the TCT training data[4] and had annotators assign CTB-style tags to the sentences. Hereafter, this test set is referred to as *TCT-Conversion-Set*, which provides us an additional dataset for POS conversion experiments. The other group of experiments aim to examine the effectiveness of newly-gained data (achieved by converting the TCT training data) when the data is used as additional training data for building POS taggers and syntactic parsers. One point worthy of note is that we actually ignore the distinctions in word segmentation standards of CTB and TCT, because for Chinese language, "although different segmentation standards exist, the differences are limited" [Low et al. 2005]. Refer to Gao et al. [2005] and Jiang et al. [2009] for the discussion of annotation conversion in the word level.

2.2 Treebank Comparison

Before describing approaches to treebank conversion, it is imperative to present a comparative study on CTB and TCT, both quantitatively and qualitatively. Actually, previous work shows that comparing treebanks is not as straightforward or as easy as expected [Rehbein and Genabith 2007]. In the following, we first give an overview of

[2]The development set is not used in this article.

[3]See http://www.cipsc.org.cn/clp2010/task2_en.htm.

[4]We sampled the 1,000 sentences from the TCT training data instead of using the TCT test directly. The reason is that we require a sample that has a near distribution as the TCT training data such that we can use the conversion accuracy on the sample to approximate the accuracy on the whole TCT training data. But to our knowledge, the TCT test set is collected from a source quite different from the TCT training data.

Table II. Some Properties of CTB and TCT

	avg.sent. length	avg.word. length	vocab. size	non-term/term. nodes
CTB	23.01	1.62	36,616	1.22
TCT	27.44	1.60	30,452	0.75

the main features of CTB and TCT, then use some statistics to quantitatively measure similarity and difference between these two treebanks.

In the POS level, CTB has 33 different tags and TCT uses 46 tag types. For some parts of speech, TCT has finer-grained annotation than CTB. For example, TCT uses five distinct tags to categorize punctuation. CTB, instead, uniformly annotates all kinds of punctuation as *PU*. Another example is the annotation of named entities. TCT distinguishes between *person name*, *location name*, *organization name*, and other named entities, whereas CTB uniformly annotates all types of named entities as *NR*. On the other hand, adjective words which are annotated as *a* in TCT might be annotated either as *JJ* or as *VA* in CTB. In summary, TCT overall has finer-grained annotation than CTB, but not for all parts of speech.

In the syntactic structure level, CTB is annotated with a much more detailed set of grammar labels. Specifically, TCT and CTB use 16 and 28 grammar label types, respectively. Some phrase types, such as *NP* (noun phrase) can find their counterparts in TCT, where noun phrases are annotated as *np*. But for some phrase types, such as *DNP* in CTB, there are no counterparts. Such consensus and difference in grammar labels are, to some extent, illustrated in Figure 1. As to hierarchical tree structure, CTB and TCT have the following distinctions.

— CTB is widely known for the preference of flat noun phrases, whereas TCT generally adopts a hierarchical annotation for noun phrases. See the noun phrase 中国 (Chinese) 传统 (traditional) 文化 (culture) in Figure 1 as an example.
— TCT uses no unary productions, but in CTB, unary productions widely exist. See different annotations for the word 我们 (our) in Figure 1.

For quantitative comparison, we utilize statistics including average sentence/word length, vocabulary size, and the ratio of non-terminal (excluding pre-terminals) vs. terminal nodes. Statistics obtained on the CTB and TCT training data are shown in Table II.

From the table, we can see that CTB and TCT have comparable average sentence/word lengthes, but differ dramatically in the vocabulary size and the ratio of non-terminal over terminal nodes. More specifically, CTB have much more word types than TCT, although they have close sentence numbers. Actually, only 23,853 word types are common in CTB (65.14%) and TCT (78.33%). The difference in the ratio of non-terminal vs. terminal nodes indicates that CTB is inclined to assign parse trees of more levels. One possible reason is that CTB uses unary productions.

Finally, we are interested in the bracketing consensus of CTB and TCT, that is, the ratio of common brackets when CTB and TCT assign syntactic trees to the same set of sentences. For this purpose, we use CTB-Conversion-Set, since this set contains both CTB-style and TCT-style parse trees for the same 348 sentences. Specifically, 53.2%(4624/8695) of unlabeled constituents of CTB-style parse trees also appear in TCT-style parse trees. With respect to TCT, the corresponding number is relatively higher, which is 69.4%(4303/6204).

3. PART-OF-SPEECH TAG CONVERSION

This section focuses on the subtask of POS conversion. Specifically, we take segmented sentences and original POS annotations as input, and the aim is to convert the POS tags to another standard.

3.1 Our Approach

For the convenience of discussion, we denote the source and target POS datasets (derived from treebanks) by D^s and D^t, respectively. The tag set of a source dataset is named as *source tag set* and correspondingly, that of a target dataset is named as *target tag set*. The source tag set and target tag set are denoted by T^s and T^t, respectively.

3.1.1 Motivation. At first glance, the POS conversion problem can be solved with some straightforward approaches. One obvious way is to build POS taggers on a target dataset which in turn assign tags to source datasets. This approach, however, fails to make use of original annotations in the source dataset which provide strongly indicative information for selecting appropriate target tags. Moreover, conversion performance of this approach depends heavily on the state of the art of POS taggers and distributional differences between source and target datasets.[5]

Another approach for solving the conversion problem is to manually construct mapping rules from T^s to T^t according to annotation guidelines. This approach is viable only if the underlying mapping function $f : T^s \rightarrow T^t$ is single-valued. That is, for each input source tag $t^s \in T^s$, there is one and only one output target tag $t^t \in T^t$. From the aforementioned comparison between TCT and CTB, we know that such a single-valued function does not exist in some cases.

By taking into account the fact that the mapping function has multiple outputs, an alternative to rigorous mapping rules is to model correspondence relationships between source and target tags in a probabilistic point of view. To this end, we opt to introduce probabilities into the mapping function, which leads to a function of the form $f : T^s \xrightarrow{P} T^t$. Here, P denotes a probability distribution on mappings. When formulating the conversion problem, the probabilistic mapping function can be defined to be of various forms, such as $P(t^t|t^s)$, $P(t^s|t^t)$, and $P(t^t, t^s)$, where $t^s \in T^s$ and $t^t \in T^t$. In order to avoid the data sparseness problem in practical application, we would like to get around the dependence of mapping probabilities on specific words. Note that probabilistic mapping functions model the relationship between source and target tags. Such information will be incorporated into a probabilistic approach to POS conversion described in the following.

3.1.2 Problem Formulation. In this work, we view automatic POS conversion from the perspective of sequence inference. Given a word sequence o with a source tag sequence s, a generative conversion framework is formalized in order to search for the optimal target sequence t^*.

$$\begin{aligned} t^* &= \arg\max_{t \in G(o)} P(t, s, o) \\ &= \arg\max_{t \in G(o)} P(t, o) P(s|t, o), \end{aligned} \tag{1}$$

where $G(o)$ represents the set of all possible target sequences of o. It is interesting to notice that $P(t, o)$ is only dependent on target data, and $P(s|t, o)$, on the other hand,

[5]By using diverse test sets, experiments in Subsection 4.2 empirically demonstrate accuracy degradation caused by differences between datasets.

bridges the tags from source and target datasets. Hereafter, $P(t, o)$ is referred to as *target component*, and $P(s|t, o)$ is referred to as *label correspondence component*.

In regard to $P(t, o)$, we can model it as any sequence labeling methods such as Hidden Markov Models (HMMs) and Conditional Random Fields (CRFs). For simplicity of implementation, we choose to use HMMs, which are defined in Equation (2).

$$\hat{t} = \arg\max_{t \in G(o)} P(t, o) = \arg\max_{t \in G(o)} P(t)P(o|t) \tag{2}$$

In this article, we build a second-order HMM introduced in Thede and Harper [1999]. For unknown words, we slightly modify the unknown-word model used in Thede and Harper [1999] by considering prefixes and suffixes of length up to two Chinese characters.

The term $P(s|t, o)$ is noteworthy because it connects original annotation s with the tag sequence t which we intend to search for. By assuming that the source tag at position i is only dependent on the word and target tag at the same position,[6] this term can be decomposed into a product of multiple word-specific terms, as follows:

$$P(s|t, o) = \prod_i P(s_i|t_i, o_i). \tag{3}$$

We back off the probability $P(s_i|t_i, o_i)$ to $P(s_i|t_i)$ in view of data sparseness problem[7] which potentially causes unreliable estimation of $P(s_i|t_i, o_i)$. Note that $\{P(s_i|t_i)\}$ are just the probabilities $\{P(t^s|t^t)\}$ introduced in the preceding subsection. Hereafter we use the term *label correspondence probability* to refer to $\{P(t^s|t^t)\}$. And correspondingly, the methods used for learning $\{P(s_i|t_i)\}$ are named *label correspondence learning*. The overall conversion model is thus formalized in Equation (4).

$$\begin{aligned} t^* \;=\; & \arg\max_{t \in G(o)} \prod_i P(t_i|t_{i-2}t_{i-1}) \times \\ & \prod_i P(o_i|t_{i-1}t_i) \times \prod_i P(s_i|t_i) \end{aligned} \tag{4}$$

We can see that the model manages to integrate information from both source and target datasets. We notice that parameter estimation for the terms $\{P(t_i|t_{i-2}t_{i-1})\}$ and $\{P(o_i|t_{i-1}t_i)\}$ can be conducted on a target dataset in the same way as conventional HMMs. So the last open problem is how to estimate label correspondence probabilities $\{P(s_i|t_i)\}$. This shall be described in detail as follows. With the parameters being available, a Viterbi algorithm with slight modifications can be used to search for the best output t^*. During the decoding phase, information derived from original annotation (in the form of $\{P(s_i|t_i)\}$) is used to guide the search procedure. Actually, this is the reason why our approach is named as *informed decoding*.

Before introducing the methods for label correspondence learning, let us examine why the relationships between source and target tags may assist in conversion. Briefly speaking, the reason is that source tags can provide strongly indicative information to conversion systems. More specifically,

— some estimated label correspondence probabilities are skewed. For example, consider the mapping from the source tag *n* (common noun) to the target tag *NN*. The label correspondence learning methods should give a high probability to the pair

[6]Whenever needed, the assumption can be relaxed by assuming that the current source tag is also dependent on preceding target tags.

[7]This claim is supported by statistics from TCT which has more than 700K words; there are approximately 12.2% word types that occur more than 10 times in this dataset.

(n, *NN*) since, with a few exceptions, people generally agree on whether a word is a noun. Such a probability pushes conversion systems to prefer *NN* to other target tags whenever the source tag n is seen.

— for the cases where mapping probabilities are not skewed, source tags help to clamp possible target tags into a small subset of T^t. For example, the word 年轻 (young) has three target tags: *JJ*, *VA*, and *NN*. When 年轻 (young) appears with the source tag a, we need only to choose between VA and JJ since the source tag a indicates that 年轻 (young) is an adjective.

3.1.3 Learning Label Correspondence. The most straightforward method to learn mapping probabilities $\{P(s_i|t_i)\}$ is to manually annotate a dataset with both source and target tags, such that the probabilities can be estimated on the dataset by simply using maximum likelihood estimation. This solution, however, has to consider the trade-off between annotation cost and estimation reliability. Specifically, manually annotating data is generally of high cost. On the other hand, the size of the newly annotated dataset should reach some minimum requirement in order to render the estimation of mapping probabilities reliable. To overcome this dilemma, we propose two fully automatic methods that do not incur any additional annotation cost.

Supervised Tagger-Based Method

The basic idea of the supervised tagger-based method is to automatically annotate a dataset with POS taggers trained on the other dataset. For example, we can train a CTB-based tagger and then use it to assign CTB-style tags to the TCT data thus each word of which is accompanied with two tags, from CTB and TCT respectively. On the newly generated dataset, label correspondence probabilities can be calculated using the following formula:

$$P(t^s|t^t) = \frac{Count(t^s, t^t)}{Count(t^t)}.$$

Clearly, the performance of the supervised tagger-based method is based on two factors: the accuracy of learned POS taggers and the size of the dataset on which label correspondence probabilities are estimated.

The dependence of performance on the accuracy of POS taggers is a potential disadvantage of this method, since the accuracy of POS taggers might be unsatisfactory in some practical situations. Generally speaking, the accuracy of POS taggers is determined jointly by the state of the art of POS tagging and the gap between the distributions of training and testing data. Firstly, for some languages, for example, English and Chinese, the problem of POS tagging has been extensively studied and the state-of-the-art systems are able to achieve accuracy scores higher than, say 96% [Huang et al. 2007; Thede and Harper 1999]. However for some languages such as Hebrew [Bar-haim et al. 2008], comparatively there is much room left for further performance improvement. Secondly, conventional evaluation of POS tagging accuracy generally uses training and testing data that are different portions of the same corpus, say CTB. A performance degradation tends to exist when the distribution of training data differs from that of testing data.

Common Word-Based Method

The supervised tagger-based method relates two datasets indirectly through POS taggers. At this point, consider how to approach this problem by directly using common words across datasets. A word w is said to be common if it appears in both datasets. The label correspondence learning method that uses common words is named as *common word-based method*.

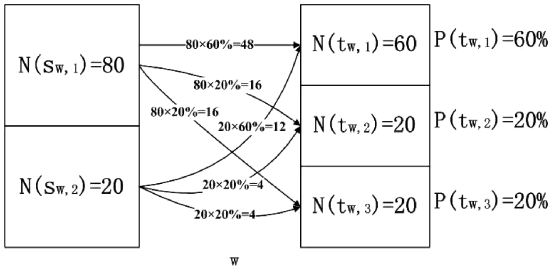

Fig. 2. An illustrating example of common word-based correspondence learning.

We first give an intuitive explanation of this method with a specific common word. The word 解决/solve has tags *VV* (192) and *NN* (39) in CTB and has tags v (178) and vN (9) in TCT. Here the numbers are the tag frequencies of the word in CTB and TCT, respectively. Assume that the source and target datasets have similar distributions. In other words, if a word acts as, say a verb, most of the time in one dataset, then it is also likely to be a verb in the other dataset. Given the assumption and the frequency counts, it is natural to relate *VV* with v rather than with vN. Pairs in $\{VV, NN\} \times \{v, vN\}$ should be given different weights (real numbers) in order to indicate diverse correlation degree of the words in each pair. Figure 2 is a synthetic example used to formally describe the process of giving such weights. Suppose there is a common word w which has two tags in the source dataset and three tags in the target dataset. $\{N(\cdot)\}$ refer to the numbers of co-occurrences of the word and its tags, as shown in the figure. The first step is to calculate the prior distribution using the counts of the target tags of the word w. Then this distribution is used to partition the counts on the source dataset side. In this way, we can get weights for each member in the Cartesian product $\{s_{w,1}, s_{w,2}\} \times \{t_{w,1}, t_{w,2}, t_{w,3}\}$. These two steps amount to the calculation using the following formula:

$$W(s_{w,i}, t_{w,j}) = N(s_{w,i}) \times \frac{N(t_{w,j})}{\sum_k N(t_{w,k})}.$$

We can see that the weights $\{W(s_{w,i}, t_{w,j})\}$ are invariant relative to the size of target dataset if the prior distribution keeps stable.

In order to achieve accurate estimation, we favor to sum over all common words to get accumulative weights of tag pairs in $T^s \times T^t$. Algorithm 1 details the procedure of the common word-based method. In brief, the algorithm is modular with two components. The first two steps collect common words and the frequency counts of words paired with corresponding tags in source and target datasets, respectively. The third step is the aforementioned procedure which conducts calculation on each common word. The results of all common words are aggregated. The algorithm finally returns a $|T^s| \times |T^t|$ matrix mp whose elements correspond to the weights of the pairs $(t^s \in T^s, t^t \in T^t)$. From the statistics contained in the matrix we estimate mapping probabilities using the formula:

$$P(t^s|t^t) = \frac{mp_{i,j}}{\sum_k mp_{k,j}},$$

where i denotes the row index of t^s and j denotes the column index of t^t. We can see that the algorithm has linear-time complexity of $O(C_1 \cdot C_2 \cdot |L|)$, where C_1 represents the average number of source tags for each common word and the C_2 represents that number of target tags.

Algorithm 1: Common Word-Based Label Correspondence Learning

Argument: D^t {dataset annotated with target tags}
D^s {dataset annotated with source tags}

Returns: mp{$|T^s| \times |T^t|$ mapping matrix.}

Steps:

1. build common word list l in D^s and D^t.

2. for each word w in l, **do**
  for each possible tag t of w in D^t, **do**
    T-COUNT{w}{t} = count of $\langle w, t \rangle$ in D^t.
  for each possible tag t of w in D^s, **do**
    S-COUNT{w}{t} = count of $\langle w, t \rangle$ in D^s.

3. for each word w in l, **do**
  T-DIST = calculate-prior-distribution(T-COUNT{w}).
  for each tag t^s in S-COUNT{w}, **do**
    for each tag t^t in T-DIST, **do**
      mp{t^s}{t^t} += S-COUNT{w}{t^s} $\times$ T-DIST{t^t}

4. return (mp)

Subroutine: calculate-prior-distribution:
Given counts of tags of a word, this subroutine calculates and returns the prior distribution.

Note that Algorithm 1 first calculates prior distributions on the target side and then uses them to partition counts from the source dataset. We can propose a variant algorithm by reversing the direction, an algorithm that calculates prior distributions on the source side.

3.2 Experiments

For all the experiments in the POS level, accuracy is used as the performance metric which is defined to be the ratio of correctly assigned POS tags over all POS tags in a test set.

3.2.1 Systems for Conversion. As mentioned above, the naive approach to POS conversion is to utilize POS taggers which make no use of original annotations in the source dataset. In this article, we use CRF and HMM-based POS taggers as baseline systems. Hereafter, these two baselines are referred to as *CRF-D* and *HMM-D*, respectively, where the suffix *D* represents *direct POS tagging*. For the CRF-based POS tagger, we use the implementation *crfsgd*,[8] and feature templates are listed in Table III. The implementation of HMMs is inspired by the work in Thede and Harper [1999], with some modifications to the unknown-word model, as described in Section 3.1.

To realize the informed decoding approach, we always use a second-order HMM to model the target component in Equation (1), parameters of which are estimated on a target dataset only. But we have multiple versions of informed decoding methods, each of which corresponds to a label correspondence learning method. For the supervised

[8]See http://leon.bottou.org/projects/sgd/.

Table III. Feature Templates for CRF-Based Chinese POS Tagger

Feature Templates	Description
a) $w_n(-2, -1, 0, 1, 2)$	unigram of words
b) $w_n w_{n+1}(-2, -1, 0, 1)$	bigram of words
c) $w_{-1} w_1$	preceding and following words
d) $w_n w_{n+1} w_{n+2}(-2, -1, 0)$	trigram of words
e) $Len(w_0)$	character number of current word
f) $PreOne(w_0)$	first character of current word
g) $PreTwo(w_0)$	first two characters of current word
h) $SufOne(w_0)$	last character of current word
i) $SufTwo(w_0)$	last two characters of current word
j) Other morphological features	is punctuation, number, foreign word

tagger-based method, we need to decide on two factors: 1) machine learning algorithms for implementing POS taggers, and 2) the dataset for training POS taggers. Thus we have four versions of supervised tagger-based method: HMM-T, HMM-S, CRF-T, and CRF-S. Here the suffixes, T (target) and S (source), indicate where the training data for POS taggers comes from. For example, HMM-T refers to the one that trains a POS tagger on the training data from the target dataset (CTB) and calculates mapping probabilities on the training data from the other dataset (TCT). Regarding the common word-based method, CW-T refers to the one that calculates prior distributions on the target dataset, and CW-S represents the reverse case.

3.2.2 Experiments on Conversion. Table IV shows the POS conversion results on CTB-Conversion-Set, where TCT-style tags in this set are converted to CTB-style tags. Since in practical cases, available target dataset might be in a variety of sizes, we mimic such situations by using portions (first row in Table IV) of the CTB training data as target datasets. For example, 40% means 7,240 sentences (of 18,100) in the CTB training data are used as a target dataset. To relieve the effect of ordering, we randomly shuffle parse trees in the CTB training data before the data is used.

From the results we can see: 1) the state of the art of Chinese POS tagging with CRFs and second-order HMMs on CTB5.1, 2) the efficacy of the informed decoding methods on POS conversion, especially when a target dataset is small in size, and 3) diverse versions of label correspondence learning methods actually achieve comparable results. As to the third point listed above, by comparing the results of HMM-S (HMM-T) and CRF-S (CRF-T), we can infer that supervised tagger-based methods for learning label correspondence probabilities are to some degree insensitive to the choice between CRFs and HMMs for label correspondence learning.

It is interesting to compare our methods with a recently proposed conversion method, Jiang et al. [2009]. Based on the CRF POS tagger described above, Jiang et al. [2009] achieves an accuracy of 96.15% on CTB-Conversion-Set, when the whole CTB training data is used. We can see that this result is better (+0.26%) than the best result (95.89%) our methods can achieve. But it is worth noting that if we formulate the target component in Equation (1) with CRF models, our methods are conceivable to get better results.

We also conduct experiments to examine how the size of dataset on which label correspondence probabilities are estimated affects conversion accuracy. For this purpose, we consistently estimate target probabilities on the whole CTB training data. That is, parameters of the target component are kept unchanged in the experiments. For label correspondence learning, we use varying portions of the CTB training data as the datasets on which label correspondence probabilities are estimated. Label

Table IV. POS Conversion Accuracy on CTB-Conversion-Set

Ratio	20%	40%	60%	80%	100%
HMM-D	87.82	88.41	89.82	92.34	93.09
CRF-D	90.17	91.02	92.15	94.66	95.02
HMM-T	93.15	94.23	94.59	95.52	95.71
HMM-S	93.17	94.21	94.64	95.55	95.78
CRF-T	93.36	94.43	94.71	95.68	95.87
CRF-S	**93.57**	**94.39**	**94.87**	**95.73**	**95.89**
CW-T	93.13	94.15	94.59	95.47	95.64
CW-S	93.17	94.20	94.58	95.53	95.61

Table V. Comparative Study of Diverse Label Correspondence Learning Methods

Ration	20%	40%	60%	80%	100%
HMM-S	95.71	95.76	95.77	95.77	95.78
CRF-S	95.83	95.86	95.87	95.89	95.89

correspondence learning methods HMM-S and CRF-S are applied. The results are shown in Table V.

From the results we can see that estimating label correspondence probabilities actually requires a limited size of data (20% data can achieve very close results as 100% data). Considering results in both Table IV and Table V, we can conclude that diverse label corresponding methods can provide reliable estimation for label correspondence probabilities. One possible further step to achieve better conversion results shall lie in better modeling of the target component in Equation (4), for example, using CRFs. This will be our further work.

Experimental results presented in Table IV demonstrate effectiveness of the informed decoding approach, though, the results also release a fact that advantage of informed decoding methods trails off with the increase of the size of a target dataset. But it is worth clarifying the following points.

— For informed decoding methods, we use a relatively simple model for the target component, but CRF-D instead incorporates much richer contextual features. Thus, in some sense, we compared CRF-D and informed decoding methods unfairly.
— The CTB training data and CTB-Conversion-Set have the same feature space and have the same distribution. In practical application which we are more interested in, the data to be converted, however, might have a different distribution from a target dataset.

In order to empirically study the second point described above, we evaluate conversion accuracy on TCT-Conversion-Set, by using portions of the CTB training data as target data. In contrast to CTB-Conversion-Set, sentences of TCT-Conversion-Set are collected from a different source. Moreover, compared to CTB-Conversion-Set, TCT-Conversion-Set has a higher OOV rate (8.7% vs. 3.5%).[9] The results are shown in Table VI.

[9]The OOV rate of a set is defined to be the ratio of word tokens that do not appear in the CTB training data over all the word tokens in the set.

Table VI. POS Conversion Accuracy on TCT-Conversion-Set

Ratio	20%	40%	60%	80%	100%
HMM-D	85.67	86.66	87.23	88.01	88.23
CRF-D	87.28	87.63	87.72	88.24	88.24
HMM-T	93.15	94.19	94.58	95.42	95.57
HMM-S	93.20	94.24	94.69	95.62	95.71
CRF-T	**93.52**	94.37	**94.77**	95.50	95.59
CRF-S	93.31	**94.38**	94.65	**95.63**	**95.74**
CW-T	93.09	94.13	94.51	95.38	95.43
CW-S	93.11	94.18	94.53	95.43	95.48

Table VII. Error Types and Statistics of the Results Achieved by CRF-S on CTB-Conversion-Set

POS	#Word	Ratio
NN	104	31.52%
DEC	54	16.36%
VV	34	10.36%
JJ	34	10.36%
AD	24	7.27%
DEG	21	6.36%

By comparing results in Table VI and Table IV, we can see that HMM-D and CRF-D achieve lower accuracy on TCT-Conversion-Set. Moreover, in regard to HMM-D and CRF-D, performance improvements achieved by enlarging training data trail off. On the other hand, it is of interest to notice that informed decoding methods achieve comparable results on CTB-Conversion-Set and TCT-Conversion-Set.

Error Analysis

We manually analyzed conversion results on CTB-Conversion-Set and found that there are two major group of errors. The particles 的(of) and 了(already) incur the biggest portion of conversion errors. This is attributed to the locality of information on determining tags for these words. For example, according to the CTB annotation standard, the choice between DEG and DEC for the word 的significantly depends on the phrase type immediately preceding it instead of words (and their POS) surrounding it.

Another source of errors originates from annotation disagreement between *noun* and *verb*. For some words, such as words 改革(reforming) 开放(opening-up) are always annotated as verbs in TCT, but as nouns in CTB. There are other disagreements. For example, CTB labels 国务院(the State Coucil) as a named entity but in TCT, this word is a common noun. Among errors induced from annotation disagreement, ambiguity between noun and verb accounts for the majority.

To present error types and their corresponding statistics, we analyze the result of CRF-S on CTB-Conversion-Set. There are 330 words in total which are mistakenly tagged. Table VII shows the statistics of the top six error types.

3.2.3 Experiments on POS Tagging. In the experiments presented here, we study how new data obtained through POS conversion can help to boost the accuracy of POS

Table VIII. Performance Results of HMMs and CRFs Taggers Using Enlarged Training Data

HMM-based Tagger					
Ratio	20%	40%	60%	80%	100%
CTB-Conver-Set	90.12	90.65	91.00	92.16	92.61
TCT-Conver-Set	88.34	88.48	88.83	88.32	88.42
CRF-based Tagger					
Ratio	20%	40%	60%	80%	100%
CTB-Conver-Set	91.78	92.53	93.18	94.45	95.02
TCT-Conver-Set	88.24	88.85	88.87	88.57	89.01

taggers. Thus we first convert the TCT training data[10] into the CTB standard and then combine the whole of newly gained data with portions of the CTB training data. For POS conversion, we applied the *CRF-S* method which achieves the best results in the previously conducted experiments. The results achieved with combined training data are reported on both CTB-Conversion-Set and TCT-Conversion-Set, as shown in Table VIII.

By comparing results against the baseline results shown in Table IV and V, we can see that new data consistently improve the accuracy on TCT-Conversion-Set. With respect to CTB-Conversion-Set, the situation is a bit complicated: the new data improves accuracy when original training data is small in size (20%, 40%, and 60%), but instead hurts the performance when the size of original training data is further increased. One possible reason is that when original training data suffice to achieve a high accuracy, incorporating additional disparate data might achieve negative effect.

4. SYNTACTIC STRUCTURE CONVERSION

This section moves to the subtask of converting syntactic structures. Note that for the conversion of syntactic structures, original TCT-style parse trees together with CTB-style POS tags (obtained through POS conversion) are available as input. In the following, we first motivate our approach with a specific example and then describe details of the approach to syntactic structure conversion. Finally, experimental results are presented.

4.1 Our Approach

Our informed decoding approach to converting syntactic structure proceeds in two steps: 1) build a parser on the target treebank (CTB); 2) apply the parser to decode sentences in the source treebank (TCT) with the aid of information derived from the source treebank. For convenience, parse trees in the source treebank are referred to as *source trees* and correspondingly, trees from the target treebank are referred to as *target trees*. Moreover, a parser built on the target treebank is referred to as *target parser*. Following is the motivation of our approach to syntactic structure conversion.

4.1.1 Motivation. The discussion in *treebank comparison* section shows that discrepancies between CTB and TCT do exist. So it shall be interesting to show why original

[10]The sentences in TCT-Conversion-Set are excluded.

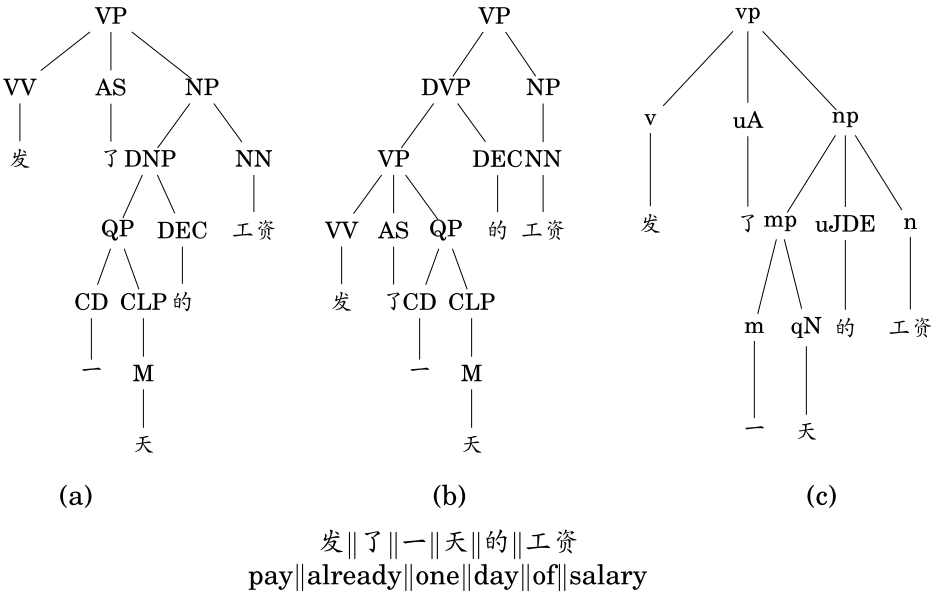

Fig. 3. Tree fragments of words 发了一天的工资: (a) and (b) show two plausible tree fragments of the words using the CTB standard; (c) shows a tree fragment of the TCT standard which has the same interpretation as (a).

annotations in a source treebank is helpful in treebank conversion. Figure 3 provides an illustrating example. The figure depicts three tree fragments for the Chinese words 发*(pay)* 了*(already)* 一*(one)* 天*(day)* 的*(of)* 工资*(salary)*, among which Figure 3(a) and Figure 3(b) are tree fragments of the CTB standard and Figure 3(c) is a tree fragment of the TCT standard. From the figure, we can see that these Chinese words actually have (at least) two plausible interpretations of the meaning. In Figure 3(a), the words mean *pay salary for one-day work* while in Figure 3(b), the words mean *spend one day on paying salary*. If no additional information is provided, we shall claim that both interpretations are rational. Due to such ambiguity, it is difficult for a parser trained on CTB to assign an appropriate syntactic tree to the words. However, if we consider above disambiguation problem in the scenario of treebank conversion and suppose Figure 3(c) is a source tree to be converted, then Figure 3(b) will be rejected since it conflicts with Figure 3(c) as to tree structures. Note that syntactic structures reflect underlying sentence meaning. On the other hand, although Figure 3(a) also has (minor) differences in tree structures from Figure 3(c), it is preferred as the conversion result.[11] From the example we can get inspired by the observation that original annotation in a source treebank is informative and necessary to converting parse trees in a source treebank.

Now we are faced with two questions: how to formalize information implicit in original annotations in a source treebank and how to make use of such information. As an answer to these questions, Wang et al. [1994] proposed a selecting-from-k-best approach where source trees are used to select one parse tree as the conversion result from each k-best list generated by a target parser. There, the selected tree is the one having maximum bracketing consensus with the corresponding source tree. In

[11]Note that we don't deny existence of annotation distinctions between the treebanks, but we aim to make use of what they both agree on. We assume that consensus is the majority.

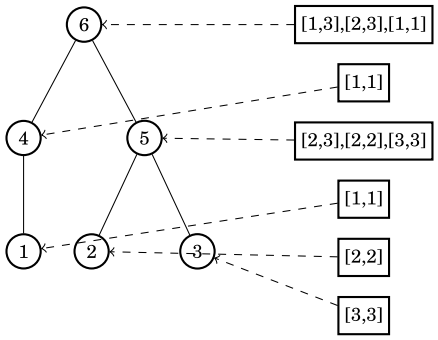

Fig. 4. Constituent set of a synthetic parse tree.

this article, we use similar formalization of information derived from source trees, but, instead of select an "optimal" tree from each k-best list, we incorporate the information into the parsing phase. The underlying motivation is twofold.

— The decoding phase of a parser is essentially a search process. Due to the extreme magnitude of searching space, pruning of search paths is practically necessary. If reliable information is provided to guide the pruning of search paths, more efficient parsing and better results are expected.
— Selecting-from-k-best works on the basis of k-best lists. Unfortunately, we often see very few variations in k-best lists. For example, 50-best trees present only 5 to 6 variations [Huang 2008]. The lack of diversities in k-best lists makes information from the source treebank less effective in selecting parse trees. By contrast, incorporating such information into the decoding phase makes the information affect the whole parse forest.

4.1.2 Formalization of Information from Source Treebank. In this article, information from a source treebank translates into two strategies which help a target parser to prune illegal partial parse trees and to rank legal partial parse trees higher. Following are the two strategies.

— Pruning strategy. Despite distinctions existing between annotation standards of the source and target treebank, the source treebank indeed provides treebank conversion with indicative information on bracketing structures and grammar labels. So when a partial parse tree is generated, it should be examined against the corresponding source tree. Unless the partial parse tree does not conflict with any constituent in the source tree, it should be pruned out.
— Rescoring strategy. In practice, decoding is often a local optimal search process. In some cases, even if a gold parse tree exists in the parse forest, parsers may fail to rank it to the top position. Rescoring strategy is used to increase scores for partial parse trees which are confidently thought to be valid.

Pruning Strategy

The pruning strategy used for treebank conversion is based on the concept of *conflict* which is defined in two dimensions: structures and grammar labels. Since a tree structure can be equivalently represented as a set of its spans (intervals of word indices) set, we can check whether two trees conflict by checking their spans. See Figure 4 for

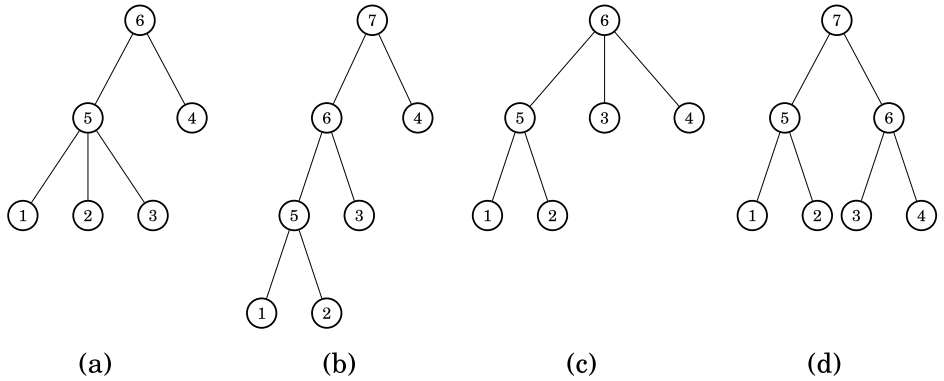

Fig. 5. Illustrating example of the concept of *conflict*: (a) and (b) are compatible (not conflict); (a) conflicts with (c) (condition 1) and (d) (condition 2).

an illustration of spans of a tree. Following are criteria determining whether two trees conflict in their structures.

— If one node in tree A is raised to be a child of the node's grandfather in tree B, and the grandfather has more than two children, then tree A and tree B conflict in structures.
— If tree A has a span $[a, b]$ and tree B has a span $[m, k]$, and these two spans satisfy the condition of either $a < m \leq b < k$ or $m < a \leq k < b$, then tree A and B conflict in structures.

Figure 5 illustrates criteria mentioned above, where Figure 5(a) is compatible (not conflict) with Figure 5(b) although they have different structures. But Figure 5(a) conflicts with Figure 5(c) (according to criterion 1; node 3 is raised) and (d) (according to criterion 2).

For the dimension of grammar labels, we manually construct a mapping between label sets (excluding POS tags) of source and target treebanks. Such a mapping is frequently a many-to-many mapping. Two labels are said to be conflicting if they are from different label sets and they cannot be mapped.

By combining these two criteria, two parse trees (of different standards) which yield the same sentence are said to be conflicting if they conflict in both structures and labels. Note that we describe pruning strategy for the case of two parse trees. In informed decoding process, this strategy is actually applied to every partial parse tree generated during decoding.

Rescoring Strategy

As mentioned above, despite that the pruning strategy helps in improving conversion accuracy, we are faced with the problem of how to rank valid parse trees higher in a parse forest. To solve the problem, we adjust the scores of partial parse trees which are considered to be confidently "good". The criteria which is used to judge "goodness" of a partial parse are listed as follows.

— The partial parse tree can find in the source tree a constituent that has the same structure as it.
— When the first criterion is satisfied, grammar categories of this partial parse should not conflict with the grammar categories of its counterpart.

Algorithm 2 CKY-style decoding

Argument: a parsing decoder
a sentence to be parsed and corresponding
source tree

Begin

Steps:

1. initialization steps

2. for span from 2 to sentence_length **do**
  for start from 1 to (sentence_length-span+1) **do**
    end := (start + span - 1)
    for each edge *e* for span [start, end] **do**
      generate(*e*, **start, end**)
      prune(*e*, **start, end**)
      rescore(*e*, **start, end**)
      add_edge(*e*, **start, end**)

End

Subroutine:

generate: generates an edge which belongs to the span [start, end].

prune: apply *pruning strategy* to check whether the edge should be pruned.

rescore: apply *rescoring strategy* to weight the edge.

add_edge: add the edge into *chart*.

In practice, we use a parameter λ to adjust the score.

$$P_{new}(e) = \lambda * P(e) \tag{5}$$

Here e represents any partial tree that is rescored, and $P(e)$ and $P_{new}(e)$ refer to original and new scores, respectively.

4.1.3 Parsing Model. Theoretically all parsing models are applicable in informed decoding, but we prefer to adopt a CKY-style parser for two reasons: CKY style parsers are dynamically bottom-up and always have edges (or parsing items) belonging to the same span stacked together in the same chart[12] cell. The property of CKY-style parsers being dynamically bottom-up can make the pruning strategy efficient by avoiding rechecking subtrees that have already been checked. The property of stacking edges in the same chart cell makes CKY-style parsers easily portable to the situation of informed decoding. In this article, the Collins parser [Collins 1999] is used. Algorithm 2 presents the extended version of the decoding algorithm used in the Collins parser. What the algorithm needs to do is to generate edges for each span. And before edges are allowed to enter the chart, pruning conditions should be checked in *prune* subroutine and rescoring should be conducted in *rescore* subroutine by consulting the corresponding source tree.

[12]Data structure used to store paring items that are not pruned.

Table IX. Conversion Accuracy with Varying Size of Target Training Data

Ratio	20%	40%	60%	80%	100%
DP	73.12	75.17	79.38	80.54	81.25
Wang94B	74.88	76.65	77.91	81.39	82.24
Wang94C	74.05	75.88	77.21	80.73	81.49
This article	**82.65**	**82.93**	**83.25**	**84.66**	**84.21**

4.2 Experiments

Analogous to the experiments in the POS level, experiments presented here also serve two respective purposes: evaluate conversion accuracy in the syntactic structure level and examine the effect of adding newly generated data as additional training data for syntactic parsing. For experiments on conversion, we convert TCT-style parse trees in CTB-Conversion-Set to the CTB standard and evaluate conversion results against the CTB-style parse trees in the same set. In all the experiments, whenever POS conversion is needed, we use the CRF-S method which achieves the best results among all the informed decoding methods for POS conversion. Concerning the parameter λ in Equation (5), we used the value of 0.3 which achieves the best conversion result on CTB-Conversion-Set. Note that we tune the parameter directly on the test set because we are interested in the best conversion accuracy we can achieve on the dataset. After the parameter tuning we can use the optimal parameter value when we convert the whole TCT training data. For performance evaluation, *bracketing F1* is used, which is provided by the EVALB program.[13]

4.2.1 Experiments on Conversion. In the experiments, we use two representative baseline systems. One, named as *directly parsing (DP)*, assigns CTB-style parse trees to the sentences in CTB-Conversion-Set directly with a Collins parser trained on target data, and the other is the method proposed in Wang et al. [1994]. For the latter baseline, we use two parsers, Berkeley parser [Petrov et al. 2006] and Collins parser [Collins 1999] to generate k-best lists. Berkeley parser is able to provide k-best lists with high oracle scores than Collins parser does [Zhang et al. 2009]. Hereafter, implementations of the approach in Wang et al. [1994] with a Berkeley parser and Collins parser are referred to as *Wang94B* and *Wang94C*, respectively. Specifically speaking, Wang94B (Wang94C) proceeds in two steps: 1) use Berkeley parser (Collins parser) to produce k-best lists of CTB-style parse trees for raw sentences in CTB-Conversion-Set and 2) select a parse tree from each k-best list with respect to TCT-style annotation in CTB-Conversion-Set. Here we set k to 50. Table IX reports F1 scores of the baseline systems and our informed decoding approach with varying size of target training data. As the experiments in the POS level, the first row of the table represents fractions of the CTB training data which are used as target datasets.

From the results, we can see that our approach performs significantly better than DP, Wang94C, and Wang94B. Specifically, when 100% CTB training data is used as target training data, 2.96% absolute improvement is achieved. When the size of target training data decreases, absolute improvements of our approach over baseline systems are further enlarged. More interestingly, decreasing in target training data only results in marginal decrement in conversion accuracy of our approach. This is of significant importance in situations where a target treebank that we can acquire is small in size.

[13]See http://nlp.cs.nyu.edu/evalb.

Table X. Conversion Accuracy on Different Span Lengths

Span Length	2	4	6	8	10
Wang94B	82.47	**83.94**	**80.75**	**77.78**	**71.73**
This article	**83.75**	82.91	79.86	77.32	70.71
Span Length	12	14	16	18	20
Wang94B	**75.26**	68.01	77.26	70.80	76.58
This article	71.81	**75.06**	**86.27**	**80.03**	**80.01**

Table XI. Conversion Results with Respect to Different Categories

Category	ADJP	VCD	CP	DNP	ADVP
Wang94B	79.63	57.12	65.41	84.78	91.60
This article	**88.02**	**66.61**	**71.64**	**88.28**	**93.48**

In order to evaluate the accuracy of conversion methods on different span lengths, we compare the results of Wang94B and informed decoding produced by using 100% CTB training data. Note that we only use Wang94B to participate the comparison since it is a relatively stronger baseline. Table X shows the statistics.

From the results we can see that, having span 12 as an exception, our approach performs significantly better on long spans and achieves marginally lower accuracy on small ones. But notice that the informed decoding approach is implemented on the base of Collins parser and that Wang94B works on the basis of Berkeley parser. Taking the performance gap of the Collins parser and the Berkeley parser into account, we actually can conclude that on small spans, our approach is able to achieve results comparable with or even better than Wang94B. We can also infer from the observation that our approach can outperform Wang94B when converting parse trees which yield long sentences.

Another line of analysis is to compare the results of Wang94B and our approach with respect to individual grammar categories. Table XI lists five grammar categories in which our approach achieves most improvements. For categories *NP* and *VP*, absolute improvements are 1.1% and 1.4% respectively. Take into account large amounts of instances of *NP* and *VP*, the improvements are also quite significant.

4.2.2 Experiments on Parsing. Before doing the experiments of parsing, we first converted the whole TCT training data using 100% CTB training set as target training data. Using the newly-gained data only as training data for the Collins parser, we can get F1 score 75.4% on the CTB test data (using gold POS). We can see that the score is much lower than the accuracy achieved by using the CTB training data (75.21% vs. 82.04%). Possible reasons that result in lower accuracy include: 1) divergences in word segmentation standards between TCT and CTB, 2) divergences of domains of TCT and CTB, and 3) conversions errors in newly-gained data. Although the newly-gained data cannot act as a replacement of the CTB training data, we would like to use it as additional training data besides the CTB training data. Following experiments aim to examine effectiveness of the newly-gained data when used as additional training data.

In the first parsing experiment, the TCT corpus is converted using portions of the CTB training data. As in the conversion experiments, parse trees in the CTB training data are randomly ordered before splitting of the training set. For each portion, newly gained data together with the portion of the CTB training data are used to train a new parser. Evaluation results on the CTB test data are presented in Table XII.

Table XII. Parsing Accuracy with New Data Included

Ratio	20%	40%	60%	80%	100%
Collins	75.74	77.65	79.43	81.22	82.04
Collins+	**78.81**	**79.46**	**80.01**	**81.72**	**82.35**

Table XIII. Parsing Accuracy with New Data Included

# of Added Data	2k	4k	6k	8k
Labeled Data	78.51	79.52	80.01	81.37
Auto Data	78.21	79.07	79.83	79.63

Here in Table XII, the first row represents ratios of parse trees from the CTB training data. For example, 40% means the first 40% parse trees in the CTB training data are used. The *Collins* row represents the results of only using portions of the CTB training data, and the *Collins+* row contains the results achieved with enlarged training data. From the results, we find that new data indeed provides complementary information to the CTB training data, especially when the training data is small in size. But the benefits of a Collins parser gained from additional training data level out as the size of training data grows. To examine whether techniques such as corpus weighting [Niu et al. 2009] that assign different weights to training data and the additional data can further improve parsing accuracy, we increase the weight of CTB training instances when the whole CTB training data is used. Here, for simplicity of implementation, we only use integral weights. Experimental results show that the largest improvement is achieved (from 82.35% to 82.77%) when the weight is set to 5.

Another obversion from Table XII is that the parser trained on 40% CTB training data plus additional training data achieves higher accuracy than using 60% CTB training data. We incrementally add labeled training data and automatic training data respectively to 40% CTB training data. The purpose of this experiment is to see the magnitude of automatic training data which can achieve the same effect as labeled training data does. The results are depicted in Table XIII.

From the results we see that accuracy gaps between using labeled data and using automatic data get large with the increment of added data. One possible reason is that more noise is taken when more data is added. This observation further verifies that refining techniques such as corpus weighting are necessary for using automatically gained data.

5. RELATED WORK

Previous work on treebank conversion can be grouped into two categories according to whether grammar formalisms of treebanks are identical. One type focuses on converting treebanks of different grammar formalisms. Collins et al. [1999] addressed constituent syntactic parsing on Czech using a treebank converted from a Prague dependency treebank. For the purpose of treebank conversion, conversion rules are derived from head-dependent pairs and heuristic rules are also applied. Xia and Palmer [2001] compared three algorithms for conversion from dependency structures to phrase structures. The algorithms expanded each node in input dependency structures into a projection chain and labeled the newly inserted node with syntactic categories. The three algorithms differ only in heuristics adopted to build projection chains. Xia et al. [2008] automatically extracted conversion rules from a target treebank and proposed strategies to handle the case when more than one conversion rule are applicable. Instead of using conversion rules, Niu et al. [2009] proposed to convert a dependency treebank to a constituency one by using a parser trained

on a constituency treebank to generate k-best lists for sentences in the dependency treebank. Optimal conversion results are selected from the k-best lists. There are also efforts on the conversion in the reverse direction, that is, from a constituency treebank to a dependency treebank. The work in Nivre [2006] and Johansson and Nugues [2007] describes conversion methods to create English dependency-based treebanks from readily available constituency-based treebanks. Similarly, Ekeklint and Ekeklint and Nivre [2007] employed a conversion method for constructing dependency-based propbank from PropBank [Kingsbury et al. 2002].

Relatively few efforts have been put on conversion between treebanks that have the same grammar formalisms but follow different annotation standards. Wang et al. [1994] applied a similar framework as in Niu et al. [2009] to convert from a simple constituency treebank to a more informative one. The basic idea is to apply a parser built on a target treebank to generate k-best lists for sentences in the source treebank. Then, a matching metric is defined on the number of identical bracketing spans between two trees. Such a function computes a score for each parse tree in a k-best list and its corresponding parse tree in the source treebank. Finally, the parse tree with the highest score in a k-best list is selected to be the conversion result. The difference between our work and Wang et al. [1994] lies in two points: 1) instead of using trees from the source treebank to select parse trees from k-best lists, we propose to use such trees to guide the decoding phase of the parser built on the target treebank and 2) instead of integrating POS conversion into tree conversion as an implicit component, our work instead conducts treebank conversion in two syntactic levels. Concerning POS conversion only, the most related work is presented in Jiang et al. [2009]. Both Jiang et al. [2009] and our previous work [Zhu et al. 2009] build on the idea of using original annotations in a source treebank, but their underlying ideas are significantly different. Specifically, our approach, informed decoding, focuses on how to incorporate information from a source treebank into the decoding phrase, which implies that most of efforts should be put on deciding what information from a source dataset should be used and on how to modify decoding algorithms to incorporate such information. By contrast, the idea of the framework in Jiang et al. [2009] is to use the predications of a source dataset-based POS tagger as guiding features in a target dataset-based POS tagger. To easily incorporate multiple features, POS taggers are required to be discriminative. However, such an idea proposed in Jiang et al. [2009] is generic enough to be easily adapted to other tasks, such as dependency parsing [Jiang and Liu 2009]. Moreover, that idea is generic enough to be used either to convert a source treebank, as in Jiang and Liu [2009] or to improve analysis accuracy by using a source dataset as an additional resource, as in Jiang et al. [2009]. When using the framework proposed in Jiang et al. [2009], most of efforts should be put on designing guiding features extracted from the output of a source dataset-based predictor.

6. CONCLUSIONS

We proposed in this article a unified idea, informed decoding, for the task of converting treebanks of disparate annotation standards. We divided the conversion into two subtask: POS conversion and syntactic structure conversion. Experiments which evaluate conversion accuracy directly showed that our approaches significantly outperformed baseline systems. More interestingly we found that the size of target training data had limited effect on the conversion accuracy of our approaches. This is extremely important in the situations where a treebank whose standards we are interested in is limited in size.

We also added newly gained data as additional training data for building POS taggers and parsers. Experimental results demonstrated the effectiveness of new data on boosting POS tagging and parsing accuracy.

ACKNOWLEDGMENTS

We thank Chunliang Zhang for helpful discussions and the three anonymous reviewers for very constructive and detailed comments.

REFERENCES

BANKO, M. AND BRILL, E. 2001. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the Association for Computational Linguistics (ACL'07)*. 26–33.

BAR-HAIM, R., SIMA'A, K., AND WINTER, Y. 2008. Part-of-speech tagging of modern Hebrew text. *Nat. Lang. Eng. 14*, 2, 223–251.

CHARNIAK, E. 2000. A maximum-entropy-inspired parser. In *Proceedings of the North American Chapter of Association for Computational Linguistics (NAACL'00)*. 132–139.

COLLINS, M. 1999. Head-driven statistical models for natural language parsing. Ph.D. thesis. Pennsylvania University.

COLLINS, M., RAMSHAW, L., HAJIC, J., AND TILLMANN, C. 1999. A statistical parser for Czech. In *Proceedings of the Association for Computational Linguistics (ACL'99)*. 505–512.

EKEKLINT, S. AND NIVRE, J. 2007. A dependency-based conversion of propbank. In *Proceedings of FRAME*. 19–25.

GAO, J., LI, M., WU, A., AND HUANG, C.-N. 2005. Chinese word segmentation and named entity recognition: A pragmatic approach. *Comput. Linguist. 31*, 4, 531–574.

HUANG, L. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of the Association for Computational Linguistics (ACL'08)*. 586–594.

HUANG, Z., HARPER, M. P., AND WANG, W. 2007. Mandarin part-of-speech tagging and discriminative reranking. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Conference on Computational Natural Language Learning (EMNLP-CoNLL'07)*. 1093–1102.

JIANG, W. AND LIU, Q. 2009. Automatic adaptation of annotation standards for dependency parsing - Using projected treebank as source corpus. In *Proceedings of the International Conference on Parsing Technologies (IWPT'09)*. 25–28.

JIANG, W., HUANG, L., AND LIU, Q. 2009. Automatic adaptation of annotation standards: Chinese word segmentation and pos tagging - A case study. In *Proceedings of the Association for Computational Linguistics (ACL'09)*. 522–539.

JOHANSSON, R. AND NUGUES, P. 2007. Extended constituent-to-dependency conversion for english. In *Proceedings of the 16th Nordic Conference of Computational Linguistics (NODALIDA'07)*. 105–112.

KINGSBURY, P., PALMER, M., AND MARCUS, M. 2002. Adding semantic annotation to the penn treebank. In *Proceedings of the Conference on Human Language Technologies (HLT'02)*.

LOW, J. K., NG, H. T., AND GUO, W. 2005. A maximum entropy approach to Chinese word segmentation. In *Proceedings of the 5th SIGHAN Workshop (SIGHAN'05)*. 161–164.

NIU, Z.-Y., WANG, H., AND WU, H. 2009. Exploiting heterogeneous treebanks for parsing. In *Proceedings of the Association for Computational Linguistics (ACL'09)*. 46–54.

NIVRE, J. 2006. *Inductive Dependency Parsing.* Springer, Volume 34.

PETROV, S., BARRETT, L., THIBAUX, R., AND KLEIN, D. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the Association for Computational Linguistics (ACL'06)*. 433–440.

REHBEIN, I. AND GENABITH, J. 2007. Why is it so difficult to compare treebanks? Tiger and tuba-d/z revisited. In *Proceedings of the 6th International Workshop on Treebanks and Linguistic Theories (TLT'07)*.

THEDE, S. M. AND HARPER, M. P. 1999. A second-order hidden Markov model for part-of-speech. In *Proceedings of the Association for Computational Linguistics (ACL'99)*. 175–182.

WANG, J.-N., CHANG, J.-S., AND SU, K.-Y. 1994. An automatic treebank conversion algorithm for corpus sharing. In *Proceedings of the Association for Computational Linguistics (ACL'94)*. 248–254.

XIA, F. AND PALMER, M. 2001. Converting dependency structures to phrase structures. In *Proceedings of the Human Language Technology Conference (HLT'01)*.

XIA, F., BHATT, R., RAMBOW, O., PALMER, M., AND SHARMA, D. M. 2008. Towards a multi-representational treebank. In *Proceedings of the 7th International Workshop on Treebanks and Linguistic Theories (TLT'08)*. 159–170.

XUE, N., DONG CHIOU, F., AND PALMER, M. 2002. Building a large-scale annotated Chinese corpus. In *Proceedings of the Conference on Computational Linguistics (COLING'02)*. 1–8.

ZHANG, H., ZHANG, M., TAN, C. L., AND MARCUS, M. 2009. K-best combination of syntactic parsers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'09)*. 1552–1560.

ZHOU, Q. 1996. Phrase bracketing and annotating on Chinese language corpus. Ph.D. thesis. Beijing University.

ZHU, M. AND ZHU, J. 2010. Automatic treebank conversion via informed decoding. In *Proceedings of the Conference on Computational Linguistics (COLING'10)*. 1344–1352.

ZHU, M., WANG, H., AND ZHU, J. 2009. Label correspondence learning for part-of-speech annotation transformation. In *Proceedings of the Conference of Information and Knowledge Management (CIKM'09)*. 1461–1464.

Received November 2010; revised February 2011; accepted April 2011