Supervised Treebank Conversion: Data and Approaches

Xinzhou Jiang[2]*, Bo Zhang[2], Zhenghua Li[1,2], Min Zhang[1,2], Sheng Li[3], Luo Si[3]

1. Institute of Artificial Intelligence, Soochow University, Suzhou, China

2. School of Computer Science and Technology, Soochow University, Suzhou, China

{xzjiang, bzhang17}@stu.suda.edu.cn, {zhli13,minzhang}@suda.edu.cn

3. Alibaba Inc., Hangzhou, China

{lisheng.ls,luo.si}@alibaba-inc.com

Abstract

Treebank conversion is a straightforward and effective way to exploit various heterogeneous treebanks for boosting parsing accuracy. However, previous work mainly focuses on unsupervised treebank conversion and makes little progress due to the lack of manually labeled data where each sentence has two syntactic trees complying with two different guidelines at the same time, referred as *bi-tree aligned data*.

In this work, we for the first time propose the task of *supervised treebank conversion*. First, we manually construct a bi-tree aligned dataset containing over ten thousand sentences. Then, we propose two simple yet effective treebank conversion approaches (pattern embedding and treeLSTM) based on the state-of-the-art deep biaffine parser. Experimental results show that 1) the two approaches achieve comparable *conversion accuracy*, and 2) treebank conversion is superior to the widely used multi-task learning framework in multiple treebank exploitation and leads to significantly higher *parsing accuracy*.

1 Introduction

During the past few years, neural network based dependency parsing has achieved significant progress and outperformed the traditional discrete-feature based parsing (Chen and Manning, 2014; Dyer et al., 2015; Zhou et al., 2015; Andor et al., 2016). Most remarkably, Dozat and Manning (2017) propose a simple yet effective deep biaffine parser that further advances the state-of-the-art accuracy by large margin. As reported, their parser outperforms the state-of-the-art discrete-feature based parser of Bohnet and Nivre (2012) by 0.97 (93.76% – 92.79%) on the English WSJ data and 6.87 (85.38% – 78.51%) on the Chinese CoNLL-2009 data, respectively. Kindly note that all these results are obtained by training parsers on a single treebank.

Treebanks	#Tok	Grammar
Sinica (Chen et al., 2003)	0.36M	Case grammar
CTB (Xue et al., 2005)	1.62M	Phrase structure
TCT (Zhou, 2004)	1.00M	Phrase structure
PCT (Zhan, 2012)	0.90M	Phrase structure
HIT-CDT (Che et al., 2012)	0.90M	Dependency structure
PKU-CDT (Qiu et al., 2014)	1.40M	Dependency structure

Table 1: Large-scale Chinese treebanks (token number in million).

Meanwhile, motivated by different syntactic theories and practices, major languages in the world often possess multiple large-scale heterogeneous treebanks, e.g., Tiger (Brants et al., 2002) and TüBa-D/Z (Telljohann et al., 2004) treebanks for German, Talbanken (Einarsson, 1976) and Syntag (Järborg, 1986) treebanks for Swedish, ISST (Montemagni et al., 2003) and TUT[1] treebanks for Italian, etc. Table 1 lists several large-scale Chinese treebanks. In this work, we take HIT-CDT as a case study. Our next-step plan is to annotate bi-tree aligned data for PKU-CDT and then convert PKU-CDT to our guideline. For non-dependency treebanks, the straight-

* The first two (student) authors make equal contributions to this work. Zhenghua is the correspondence author.

[1] http://www.di.unito.it/~tutreeb/

Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers), pages 2706–2716
Melbourne, Australia, July 15 - 20, 2018.

forward choice is to convert such treebanks to dependency treebanks based on heuristic head-finding rules. The second choice is to directly extend our proposed approaches by adapting the patterns and treeLSTMs for non-dependency structures, which should be straightforward as well.

Considering the high cost of treebank construction, it has always been an interesting and attractive research direction to exploit various heterogeneous treebanks for boosting parsing performance. Though under different linguistic theories or annotation guidelines, the treebanks are painstakingly developed to capture the syntactic structures of the same language, thereby having a great deal of common grounds.

Previous researchers have proposed two approaches for multi-treebank exploitation. On the one hand, the *guiding-feature* method projects the knowledge of the source-side treebank into the target-side treebank, and utilizes extra pattern-based features as guidance for the target-side parsing, mainly for the traditional discrete-feature based parsing (Li et al., 2012). On the other hand, the *multi-task learning* method simultaneously trains two parsers on two treebanks and uses shared neural network parameters for representing common-ground syntactic knowledge (Guo et al., 2016).[2] Regardless of their effectiveness, while the guiding-feature method fails to directly use the source-side treebank as extra training data, the multi-task learning method is incapable of explicitly capturing the structural correspondences between two guidelines. In this sense, we consider both of them as *indirect exploitation* approaches.

Compared with the indirect approaches, treebank conversion aims to directly convert a source-side treebank into the target-side guideline, and uses the converted treebank as extra labeled data for training the target-side model. Taking the example in Figure 1, the goal of this work is to convert the under tree that follows the HIT-CDT guideline (Che et al., 2012) into the upper one that follows our new guideline. However, due to the lack

[2] Johansson (2013) applies the feature-sharing approach of Daumé III (2007) for multiple treebank exploitation, which can be regarded as a simple discrete-feature variant of multi-task learning.

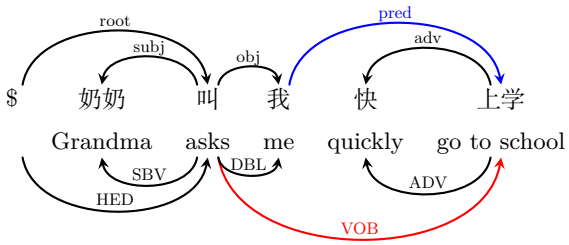

Figure 1: Example of treebank conversion from the source-side HIT-CDT tree (under) to the target-side our-CDT tree (upper).

of *bi-tree aligned data*, in which each sentence has two syntactic trees following the source-side and target-side guidelines respectively, most previous studies are based on unsupervised treebank conversion (Niu et al., 2009) or pseudo bi-tree aligned data (Zhu et al., 2011; Li et al., 2013), making very limited progress.

In this work, we for the first time propose the task of supervised treebank conversion. *The key motivation is to better utilize a large-scale source-side treebank by constructing a small-scale bi-tree aligned data.* In summary, we make the following contributions.

(1) We have manually annotated a high-quality bi-tree aligned data containing over ten thousand sentences, by re-annotating the HIT-CDT treebank according to a new guideline.

(2) We propose a pattern embedding conversion approach by retrofitting the indirect guiding-feature method of Li et al. (2012) to the direct conversion scenario, with several substantial extensions.

(3) We propose a treeLSTM conversion approach that encodes the source-side tree at a deeper level than the shallow pattern embedding approach.

Experimental results show that 1) the two conversion approaches achieve nearly the same conversion accuracy, and 2) direct treebank conversion is superior to indirect multi-task learning in exploiting multiple treebanks in methodology simplicity and performance, yet with the cost of manual annotation. We release the annotation guideline and the newly

annotated data in http://hlt.suda.edu.cn/index.php/SUCDT.

2 Annotation of Bi-tree Aligned Data

The key issue for treebank conversion is that sentences in the source-side and target-side treebanks are non-overlapping. In other words, there lacks a bi-tree aligned data in which each sentence has two syntactic trees complying with two guidelines as shown in Figure 1. Consequently, we cannot train a supervised conversion model to directly learn the structural correspondences between the two guidelines. To overcome this obstacle, we construct a bi-tree aligned data of over ten thousand sentences by re-annotating the publicly available dependency-structure HIT-CDT treebank according to a new annotation guideline.

2.1 Data Annotation

Annotation guideline. Unlike phrase-structure treebank construction with very detailed and systematic guidelines (Xue et al., 2005; Zhou, 2004), previous works on Chinese dependency-structure annotation only briefly describe each relation label with a few concrete examples. For example, the HIT-CDT guideline contains 14 relation labels and illustrates them in a 14-page document.

The UD (universal dependencies) project[3] releases a more detailed language-generic guideline to facilitate cross-linguistically consistent annotation, containing 37 relation labels. However, after in-depth study, we find that the UD guideline is very useful and comprehensive, but may not be completely compact for realistic annotation of Chinese-specific syntax. After many months' investigation and trial, we have developed a systematic and detailed annotation guideline for Chinese dependency treebank construction. Our 60-page guideline employs 20 relation labels and gives detailed illustrations for annotation, in order to improve consistency and quality.

Please refer to Guo et al. (2018) for the details of our guideline, including detailed discussions on the correspondences and differences between the UD guideline and ours.

[3] http://universaldependencies.org

Partial annotation. To save annotation effort, we adopt the idea of Li et al. (2016) and only annotate the most uncertain (difficult) words in a sentence. For simplicity, we directly use their released parser and produce the uncertainty results of all HLT-CDT sentences via two-fold jack-knifing. First, we select $2,000$ most difficult sentences of lengths $[5, 10]$ for full annotation[4]. Then, we select $3,000$ most difficult sentences of lengths $[10, 20]$ from the remaining data for 50% annotation. Finally, we select $6,000$ most difficult sentences of lengths $[5, 25]$ for 20% annotation from the remaining data. The difficulty of a sentence is computed as the averaged difficulty of its selected words.

Annotation platform. To guarantee annotation consistency and data quality, we build an online annotation platform to support *strict double annotation* and subsequent inconsistency handling. Each sentence is distributed to two random annotators. If the two submissions are not the same (inconsistent dependency or relation label), a third expert annotator will compare them and decide a single answer.

Annotation process. We employ about 20 students in our university as part-time annotators. Before real annotation, we first give a detailed talk on the guideline for about two hours. Then, the annotators spend several days on systematically studying our guideline. Finally, they are required to annotate 50 testing sentences on the platform. If the submission is different from the correct answer, the annotator receives an instant feedback for self-improvement. Based on their performance, about 10 capable annotators are chosen as experts to deal with inconsistent submissions.

2.2 Statistics and Analysis

Consistency statistics. Compared with the final answers, the overall accuracy of all annotators is 87.6%. Although the overall inter-annotator dependency-wise consistency rate is 76.5%, the sentence-wise consistency rate is only 43.7%. In other words, 56.3% $(100 - 43.7)$ sentences are further checked by a third expert annotator. This shows how

[4] Punctuation marks are ruled out and unannotated.

difficult it is to annotate syntactic structures and how important it is to employ strict double annotation to guarantee data quality.

Annotation time analysis. As shown in Table 2, the averaged sentence length is 15.4 words in our annotated data, among which 4.7 words (30%) are partially annotated with their heads. According to the records of our annotation platform, each sentence requires about 3 minutes in average, including the annotation time spent by two annotators and a possible expert. The total cost of our data annotation is about 550 person-hours, which can be completed by 20 full-time annotators within 4 days. The most cost is spent on quality control via two-independent annotation and inconsistency handling by experts. This is in order to obtain very high-quality data. The cost is reduced to about 150 person-hours without such strict quality control.

Heterogeneity analysis. In order to understand the heterogeneity between our guideline and the HIT-CDT guideline, we analyze the 36,348 words with both-side heads in the `train` data, as shown in Table 2. The consistency ratio of the two guidelines is 81.69% (UAS), without considering relation labels. By mapping each relation label in HIT-CDT (14 in total) to a single label of our guideline (20 in total), the maximum consistency ratio is 73.79% (LAS). The statistics are similar for the `dev/test` data.

3 Indirect Multi-task Learning

Basic parser. In this work, we build all the approaches over the state-of-the-art deep biaffine parser proposed by Dozat and Manning (2017). As a graph-based dependency parser, it employs a deep biaffine neural network to compute the scores of all dependencies, and uses viterbi decoding to find the highest-scoring tree. Figure 2 shows how to score a dependency $i \leftarrow j$.[5]

First, the biaffine parser applies multi-layer bidirectional sequential LSTMs (biSeqLSTM) to encode the input sentence. The word/tag embeddings $\mathbf{e}^{w_k}$ and $\mathbf{e}^{t_k}$ are concatenated as the input vector at w_k.

[5] The score computation of the relation labels is analogous, but due to space limitation, we refer readers to Dozat and Manning (2017) for more details.

Then, the output vector of the top-layer biSeqLSTM at w_k, denoted as $\mathbf{h}_k^{seq}$, is fed into two separate MLPs to get two lower-dimensional representation vectors.

$$\begin{aligned} \mathbf{r}_k^{\mathrm{H}} &= \mathrm{MLP}^{\mathrm{H}}\left(\mathbf{h}_k^{seq}\right) \\ \mathbf{r}_k^{\mathrm{D}} &= \mathrm{MLP}^{\mathrm{D}}\left(\mathbf{h}_k^{seq}\right) \end{aligned} \tag{1}$$

where $\mathbf{r}_k^{\mathrm{H}}$ is the representation vector of w_k as a head word, and $\mathbf{r}_k^{\mathrm{D}}$ as a dependent.

Finally, the score of the dependency $i \leftarrow j$ is computed via a biaffine operation.

$$\mathtt{score}(i \leftarrow j) = \begin{bmatrix} \mathbf{r}_i^{\mathrm{D}} \\ 1 \end{bmatrix}^{\mathrm{T}} \mathbf{W}^b \mathbf{r}_j^{\mathrm{H}} \tag{2}$$

During training, the original biaffine parser uses the local softmax loss. For each w_i and its head w_j, its loss is defined as $-\log \frac{e^{\mathtt{score}(i \leftarrow j)}}{\sum_k e^{\mathtt{score}(i \leftarrow k)}}$. Since our training data is partially annotated, we follow Li et al. (2016) and employ the global CRF loss (Ma and Hovy, 2017) for better utilization of the data, leading to consistent accuracy gain.

Multi-task learning aims to incorporate labeled data of multiple related tasks for improving performance (Collobert and Weston, 2008). Guo et al. (2016) apply multi-task learning to multi-treebank exploitation based on the neural transition-based parser of Dyer et al. (2015), and achieve higher improvement than the guiding-feature approach of Li et al. (2012).

Based on the state-of-the-art biaffine parser, this work makes a straightforward extension to realize multi-task learning. We treat the source-side and target-side parsing as two individual tasks. The two tasks use shared parameters for word/tag embeddings and multi-layer biSeqLSTMs to learn common-ground syntactic knowledge, use separate parameters for the MLP and biaffine layers to learn task-specific information.

4 Direct Treebank Conversion

Task definition. As shown in Figure 1, given an input sentence $\mathbf{x}$, treebank conversion aims to convert the under source-side tree $\mathbf{d}^{src}$ to the upper target-side tree $\mathbf{d}^{tgt}$. Therefore, the main challenge is how to make full use of the given $\mathbf{d}^{src}$ to guide the construction

of $\mathbf{d}^{tgt}$. Specifically, under the biaffine parser framework, the key is to utilize $\mathbf{d}^{src}$ as guidance for better scoring an arbitrary target-side dependency $i \leftarrow j$.

In this paper, we try to encode the structural information of i and j in $\mathbf{d}^{src}$ as a dense vector from two representation levels, thus leading to two approaches, i.e., the shallow pattern embedding approach and the deep treeLSTM approach. The dense vectors are then used as extra inputs of the MLP layer to obtain better word representations, as shown in Figure 2.

4.1 The Pattern Embedding Approach

In this subsection, we propose the pattern embedding conversion approach by retrofitting the indirect guiding-feature method of Li et al. (2012) to the direct conversion scenario, with several substantial extensions.

The basic idea of Li et al. (2012) is to use extra guiding features produced by the source-side parser. First, they train the source parser $Parser^{src}$ on the source-side treebank. Then, they use $Parser^{src}$ to parse the target-side treebank, leading to *pseudo* bi-tree aligned data. Finally, they use the predictions of $Parser^{src}$ as extra pattern-based guiding features and build a better target-side parser $Parser^{tgt}$.

The original method of Li et al. (2012) is proposed for traditional discrete-feature based parsing, and does not consider the relation labels in $\mathbf{d}^{src}$. In this work, we make a few useful extensions for more effective utilization of $\mathbf{d}^{src}$.

- We further subdivide their "else" pattern into four cases according to the length of the path from w_i to w_j in $\mathbf{d}^{src}$. The left part of Figure 2 shows all 9 patterns.
- We use the labels of w_i and w_j in $\mathbf{d}^{src}$, denoted as l_i and l_j.
- Inspired by the treeLSTM approach, we also consider the label of w_a, the lowest common ancestor (LCA) of w_i and w_j, denoted as l_a.

Our pattern embedding approach works as follows. Given $i \leftarrow j$, we first decide its pattern type according to the structural relationship between w_i and w_j in $\mathbf{d}^{src}$, denoted as $p_{i\leftarrow j}$. For example, if w_i and w_j are both the children of a third word w_k in $\mathbf{d}^{src}$, then $p_{i\leftarrow j}$ = "*sibling*". Figure 2 shows all 9 patterns.

Then, we embed $p_{i\leftarrow j}$ into a dense vector $\mathbf{e}^{p_{i\leftarrow j}}$ through a lookup operation in order to fit into the biaffine parser. Similarly, the three labels are also embedded into three dense vectors, i.e., $\mathbf{e}^{l_i}$, $\mathbf{e}^{l_j}$, $\mathbf{e}^{l_a}$.

The four embeddings are combined as $\mathbf{r}^{pat}_{i\leftarrow j}$ to represent the structural information of w_i and w_j in $\mathbf{d}^{src}$.

$$\mathbf{r}^{pat}_{i\leftarrow j} = \mathbf{e}^{p_{i\leftarrow j}} \oplus \mathbf{e}^{l_i} \oplus \mathbf{e}^{l_j} \oplus \mathbf{e}^{l_a} \tag{3}$$

Finally, the representation vector $\mathbf{r}^{pat}_{i\leftarrow j}$ and the top-layer biSeqLSTM outputs are concatenated as the inputs of the MLP layer.

$$\begin{aligned} \mathbf{r}^{\mathrm{D}}_{i,i\leftarrow j} &= \mathrm{MLP}^{\mathrm{D}}\left(\mathbf{r}^{seq}_{i} \oplus \mathbf{r}^{pat}_{i\leftarrow j}\right) \\ \mathbf{r}^{\mathrm{H}}_{j,i\leftarrow j} &= \mathrm{MLP}^{\mathrm{H}}\left(\mathbf{r}^{seq}_{j} \oplus \mathbf{r}^{pat}_{i\leftarrow j}\right) \end{aligned} \tag{4}$$

Through $\mathbf{r}^{pat}_{i\leftarrow j}$, the extended word representations, i.e., $\mathbf{r}^{\mathrm{D}}_{i,i\leftarrow j}$ and $\mathbf{r}^{\mathrm{H}}_{j,i\leftarrow j}$, now contain the structural information of w_i and w_j in $\mathbf{d}^{src}$.

The remaining parts of the biaffine parser is unchanged. The extended $\mathbf{r}^{\mathrm{D}}_{i,i\leftarrow j}$ and $\mathbf{r}^{\mathrm{H}}_{j,i\leftarrow j}$ are fed into the biaffine layer to compute a more reliable score of the dependency $i \leftarrow j$, with the help of the guidance of $\mathbf{d}^{src}$.

4.2 The TreeLSTM Approach

Compared with the pattern embedding approach, our second conversion approach employs treeLSTM to obtain a deeper representation of $i \leftarrow j$ in the source-side tree $\mathbf{d}^{src}$. Tai et al. (2015) first propose treeLSTM as a generalization of seqLSTM for encoding tree-structured inputs, and show that treeLSTM is more effective than seqLSTM on the semantic relatedness and sentiment classification tasks. Miwa and Bansal (2016) compare three treeLSTM variants on the relation extraction task and show that the SP-tree (shortest path) treeLSTM is superior to the full-tree and subtree treeLSTMs.

In this work, we employ the SP-tree treeLSTM of Miwa and Bansal (2016) for our treebank conversion task. Our preliminary experiments also show the SP-tree treeLSTM outperforms the full-tree treeLSTM, which is consistent with Miwa and Bansal. We did not implement the in-between subtree treeLSTM.

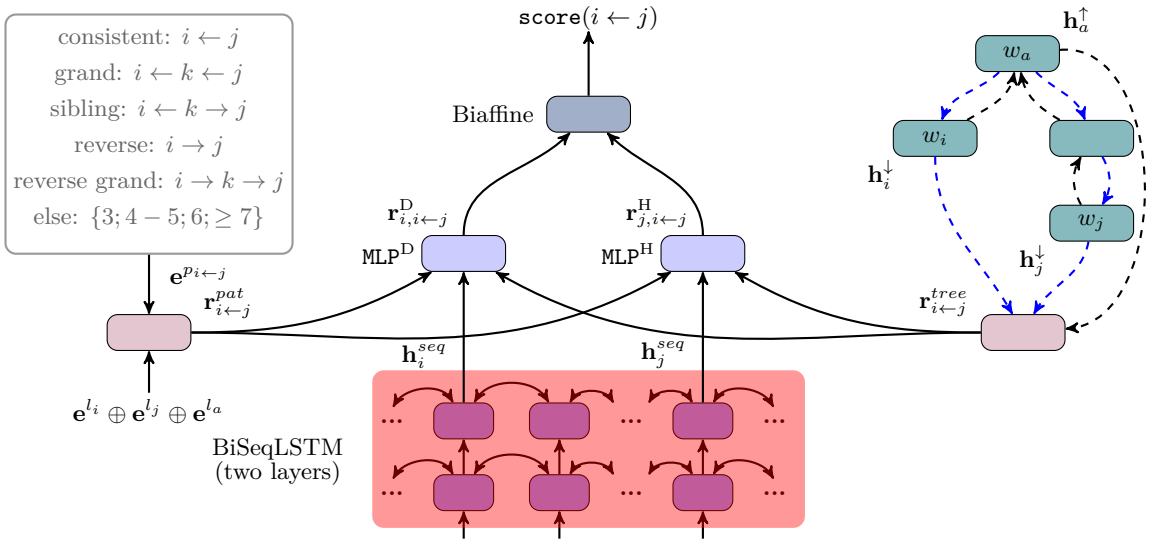

Figure 2: Computation of $\mathtt{score}(i \leftarrow j)$ in our proposed conversion approaches. Without the source-side tree $\mathbf{d}^{src}$, the baseline uses the basic $\mathbf{r}^{\mathrm{D}}_i$ and $\mathbf{r}^{\mathrm{H}}_j$ (instead of $\mathbf{r}^{\mathrm{D}}_{i,i\leftarrow j}$ and $\mathbf{r}^{\mathrm{H}}_{j,i\leftarrow j}$).

Given w_i and w_j and their LCA w_a, the SP-tree is composed of two paths, i.e., the path from w_a to w_i and the path from w_a to w_j, as shown in the right part of Figure 2.

Different from the shallow pattern embedding approach, the treeLSTM approach runs a bidirectional treeLSTM through the SP-tree, in order to encode the structural information of w_i and w_j in $\mathbf{d}^{src}$ at a deeper level. The top-down treeLSTM starts from w_a and accumulates information until w_i and w_j, whereas the bottom-up treeLSTM propagates information in the opposite direction.

Following Miwa and Bansal (2016), we stack our treeLSTM on top of the biSeqLSTM layer of the basic biaffine parser, instead of directly using word/tag embeddings as inputs. For example, the input vector for w_k in the treeLSTM is $\mathbf{x}_k = \mathbf{h}^{seq}_k \oplus \mathbf{e}^{l_k}$, where $\mathbf{h}^{seq}_k$ is the top-level biSeqLSTM output vector at w_k, and l_k is the label between w_k and its head word in $\mathbf{d}^{src}$, and $\mathbf{e}^{l_k}$ is the label embedding.

In the bottom-up treeLSTM, an LSTM node computes a hidden vector based on the combination of the input vector and the hidden vectors of its children in the SP-tree. The right part of Figure 2 and Eq. (5) illustrate the computation at w_a.

$$
\begin{aligned}
\tilde{\mathbf{h}}_a &= \sum_{k\in\mathcal{C}(a)} \mathbf{h}_k \\
\mathbf{i}_a &= \sigma\left(\mathbf{U}^{(i)}\mathbf{x}_a + \mathbf{V}^{(i)}\tilde{\mathbf{h}}_a + \mathbf{b}^{(i)}\right) \\
\mathbf{f}_{a,k} &= \sigma\left(\mathbf{U}^{(f)}\mathbf{x}_a + \mathbf{V}^{(f)}\mathbf{h}_k + \mathbf{b}^{(f)}\right) \\
\mathbf{o}_a &= \sigma\left(\mathbf{U}^{(o)}\mathbf{x}_a + \mathbf{V}^{(o)}\tilde{\mathbf{h}}_a + \mathbf{b}^{(o)}\right) \\
\mathbf{u}_a &= \tanh\left(\mathbf{U}^{(u)}\mathbf{x}_a + \mathbf{V}^{(u)}\tilde{\mathbf{h}}_a + \mathbf{b}^{(u)}\right) \\
\mathbf{c}_a &= \mathbf{i}_a \odot \mathbf{u}_a + \sum_{k\in\mathcal{C}(a)} \mathbf{f}_{a,k} \odot \mathbf{c}_k \\
\mathbf{h}_a &= \mathbf{o}_a \odot \tanh\left(\mathbf{c}_a\right)
\end{aligned}
\tag{5}
$$

where $\mathcal{C}(a)$ means the children of w_a in the SP-tree, and $\mathbf{f}_{a,k}$ is the forget vector for w_a's child w_k.

The top-down treeLSTM sends information from the root w_a to the leaves w_i and w_j. An LSTM node computes a hidden vector based on the combination of its input vector and the hidden vector of its single preceding (father) node in the SP-tree.

After performing the biTreeLSTM, we follow Miwa and Bansal (2016) and use the combination of three output vectors to represent the structural information of w_i and w_j in $\mathbf{d}^{src}$, i.e., the output vectors of w_i and w_j in the top-down treeLSTM, and the output vector of w_a

	#Sent	#Tok (HIT)	#Tok (our)
`train`	7,768	119,707	36,348
`dev`	998	14,863	4,839
`test`	1,995	29,975	9,679
`train-HIT`	52,450	980,791	36,348

Table 2: Data statistics. Kindly note that sentences in `train` are also in `train-HIT`.

	Training data	UAS	LAS
Multi-task	`train & train-HIT`	79.29	74.51
Pattern	`train`	86.66	82.03
TreeLSTM	`train`	**86.69**	**82.09**
Combined	`train`	86.66	81.82

Table 3: Conversion accuracy on `test` data.

in the bottom-up treeLSTM.

$$\mathbf{r}^{tree}_{i\leftarrow j} = \mathbf{h}^{\downarrow}_{i} \oplus \mathbf{h}^{\downarrow}_{j} \oplus \mathbf{h}^{\uparrow}_{a} \quad (6)$$

Similar to Eq. (4) for the pattern embedding approach, we concatenate $\mathbf{r}^{tree}_{i\leftarrow j}$ with the output vectors of the top-layer biSeqLSTM, and feed them into $\text{MLP}^{\text{H/D}}$.

5 Experiments

5.1 Experiment Settings

Data. We randomly select $1,000/2,000$ sentences from our newly annotated data as the `dev/test` datasets, and the remaining as `train`. Table 2 shows the data statistics after removing some broken sentences (ungrammatical or wrongly segmented) discovered during annotation. The "#tok (our)" column shows the number of tokens annotated according to our guideline. `Train-HIT` contains all sentences in HIT-CDT except those in `dev/test`, among which most sentences only have the HIT-CDT annotations.

Evaluation. We use the standard labeled attachment score (LAS, UAS for unlabeled) to measure the parsing and conversion accuracy.

Implementation. In order to more flexibly realize our ideas, we re-implement the baseline biaffine parser in C++ based on the lightweight neural network library of Zhang et al. (2016). On the Chinese CoNLL-2009 data, our parser achieves 85.80% in LAS, whereas the original tensorflow-based parser[6] achieves 85.54% (85.38% reported in their paper) under the same parameter settings and external word embedding.

Hyper-parameters. We follow most parameter settings of Dozat and Manning (2017). The external word embedding dictionary is trained on Chinese Gigaword (LDC2003T09) with GloVe (Pennington et al., 2014). For efficiency, we use two biSeqLSTM layers instead of three, and reduce the biSeqLSTM output dimension (300) and the MLP output dimension (200).

For the conversion approaches, the source-side pattern/label embedding dimensions are 50 (thus $|\mathbf{r}^{pat}_{i\leftarrow j}| = 200$), and the treeLSTM output dimension is 100 (thus $|\mathbf{r}^{tree}_{i\leftarrow j}| = 300$).

During training, we use 200 sentences as a data batch, and evaluate the model on the `dev` data every 50 batches (as an epoch). Training stops after the peak LAS on `dev` does not increase in 50 consecutive epochs.

For the multi-task learning approach, we randomly sample 100 `train` sentences and 100 `train-HIT` sentences to compose a data batch, for the purpose of corpus weighting.

To fully utilize `train-HIT` for the conversion task, the conversion models are built upon multi-task learning, and directly reuse the embeddings and biSeqLSTMs of the multi-task trained model without fine-tuning.

5.2 Results: Treebank Conversion

Table 3 shows the conversion accuracy on the `test` data. As a strong baseline for the conversion task, the multi-task trained target-side parser ("multi-task") does not use $\mathbf{d}^{src}$ during both training and evaluation. In contrast, the conversion approaches use both the sentence $\mathbf{x}$ and $\mathbf{d}^{src}$ as inputs.

Compared with "multi-task", the two proposed conversion approaches achieve nearly the same accuracy, and are able to dramatically improve the accuracy with the extra guidance of $\mathbf{d}^{src}$. The gain is 7.58 (82.09 − 74.51) in LAS for the treeLSTM approach.

It is straightforward to combine the two conversion approaches. We simply concatenate $\mathbf{h}^{seq}_{i/j}$ with both $\mathbf{r}^{pat}_{i\leftarrow j}$ and $\mathbf{r}^{tree}_{i\leftarrow j}$ before feeding into $\text{MLP}^{\text{H/D}}$. However, the "combined" model leads to no further improvement. This indicates that although the two approaches try

[6] https://github.com/tdozat/Parser-v1

	on dev		on test	
	UAS	LAS	UAS	LAS
Pattern (full)	86.73	81.93	86.66	82.03
w/o distance	86.73	81.75	86.57	81.94
w/o l_i	86.47	80.55	86.47	81.15
w/o l_j	86.55	81.69	86.45	81.76
w/o l_a	86.24	81.66	86.17	81.51
w/o labels	86.05	79.78	85.93	80.08
TreeLSTM (full)	86.73	81.95	86.69	82.09
w/o labels	86.55	80.32	86.20	80.56

Table 4: Feature ablation for the conversion approaches.

to encode the structural information of w_i and w_j in $\mathbf{d}^{src}$ from different perspectives, the resulted representations are actually overlapping instead of complementary, which is contrary to our intuition that the treeLSTM approach should give better and deeper representations than the shallow pattern embedding approach. We have also tried several straightforward modifications to the standard treeLSTM in Eq. (5), but found no further improvement. We leave further exploration of better treeLSTMs and model combination approaches as future work.

Feature ablation results are presented in Table 4 to gain more insights on the two proposed conversion approaches. In each experiment, we remove a single component from the full model to learn its individual contribution.

For the pattern embedding approach, all proposed extensions to the basic pattern-based approach of Li et al. (2012) are useful. Among the three labels, the embedding of l_i is the most useful and its removal leads to the highest LAS drop of 0.88 (82.03 − 81.15). This is reasonable considering that 81.69% dependencies are consistent in the two guidelines, as discussed in the heterogeneity analysis of Section 2.2. Removing all three labels decreases UAS by 0.73 (86.66−85.93) and LAS by 1.95 (82.03 − 80.08), demonstrating that the source-side labels are highly correlative with the target-side labels, and therefore very helpful for improving LAS.

For the treeLSTM approach, the source-side labels in $\mathbf{d}^{src}$ are also very useful, improving UAS by 0.49 (86.69 − 86.20) and LAS by 1.53 (82.09 − 80.56).

5.3 Results: Utilizing Converted Data

Another important question to be answered is whether treebank conversion can lead to higher parsing accuracy than multi-task learning. In terms of model simplicity, treebank conversion is better because eventually the target-side parser is trained directly on an enlarged homogeneous treebank unlike the multi-task learning approach that needs to simultaneously train two parsers on two heterogeneous treebanks.

Table 5 shows the empirical results. Please kindly note that the parsing accuracy looks very low, because the `test` data is partially annotated and only about 30% most uncertain (difficult) words are manually labeled with their heads according to our guideline, as discussed in Section 2.1.

The first-row, "single" is the baseline target-side parser trained on the `train` data.

The second-row "single (hetero)" refers to the source-side heterogeneous parser trained on `train-HIT` and evaluated on the target-side `test` data. Since the similarity between the two guidelines is high, as discussed in Section 2.2, the source-side parser achieves even higher UAS by 0.21 (76.20 − 75.99) than the baseline target-side parser trained on the small-scale `train` data. The LAS is obtained by mapping the HIT-CDT labels to ours (Section 2.2).

In the third row, "multi-task" is the target-side parser trained on `train & train-HIT` with the multi-task learning approach. It significantly outperforms the baseline parser by 4.30 (74.51 − 70.21) in LAS. This shows that the multi-task learning approach can effectively utilize the large-scale `train-HIT` to help the target-side parsing.

In the fourth row, "single (large)" is the basic parser trained on the large-scale `converted train-HIT` (homogeneous). We employ the treeLSTM approach to convert all sentences in train-HIT into our guideline.[7] We can see that

[7] For each sentence in `train`, which is already partially annotated, the conversion model actually completes the partial target-side tree into a full tree via constrained decoding. As shown by the results in Li et al. (2016), since the most difficult dependencies are known and given to the model, the parsing accuracy will be much higher than the traditional parsing without constraints.

	Training data	UAS	LAS
Single	`train`	75.99	70.95
Single (hetero)	`train-HIT`	76.20	68.43
Multi-task	`train & train-HIT`	79.29	74.51
Single (large)	`converted train-HIT`	**80.45**	**75.83**

Table 5: Parsing accuracy on `test` data. LAS difference between any two systems is statistically significant ($p < 0.005$) according to Dan Bikel' s randomized parsing evaluation comparer for significance test Noreen (1989).

Task	Training data	UAS	LAS
Conversion	`train`	93.42	90.49
Parsing (baseline)	`train`	89.66	86.41
Parsing (ours)	`converted train-HIT`	91.16	88.07

Table 6: Results on the fully annotated 372 sentences of the `test` data.

the single parser trained on the converted data significantly outperforms the parser in the multi-task learning approach by 1.32 (75.83 − 74.51) in LAS.

In summary, we can conclude that *treebank conversion is superior to multi-task learning in multi-treebank exploitation for its simplicity and better performance.*

5.4 Results on fully annotated data

We randomly divided the newly annotated data into train/dev/test, so the test set has a mix of 100%, 50% and 20% annotated sentences. To gain a rough estimation of the performance of different approaches on fully annotated data, we give the results in Table 6. We can see that all the models achieve much higher accuracy on the portion of fully annotated data than on the whole test data as shown in Table 3 and 5, since the dependencies to be evaluated are the most difficult ones in a sentence for the portion of partially annotated data. Moreover, the conversion model can achieve over 90% LAS thanks to the guidance of the source-side HIT-CDT tree. Please also note that there would still be a slight bias, because those fully annotated sentences are chosen as the most difficult ones according to the parsing model but are also very short ([5, 10]).

6 Conclusions and Future Work

In this work, we for the first time propose the task of supervised treebank conversion by constructing a bi-tree aligned data of over ten thousand sentences. We design two simple yet effective conversion approaches based on the state-of-the-art deep biaffine parser. Results show that 1) the two approaches achieves nearly the same conversion accuracy; 2) relation labels in the source-side tree are very helpful for both approaches; 3) treebank conversion is more effective in multi-treebank exploitation than multi-task learning, and achieves significantly higher parsing accuracy.

In future, we would like to advance this work in two directions: 1) proposing more effective conversion approaches, especially by exploring the potential of treeLSTMs; 2) constructing bi-tree aligned data for other treebanks and exploiting all available single-tree and bi-tree labeled data for better conversion.

Acknowledgments

The authors would like to thank the anonymous reviewers for the helpful comments. We are greatly grateful to all participants in data annotation for their hard work. We also thank Guodong Zhou and Wenliang Chen for the helpful discussions, and Meishan Zhang for his help on the re-implementation of the Biaffine Parser. This work was supported by National Natural Science Foundation of China (Grant No. 61525205, 61502325 61432013), and was also partially supported by the joint research project of Alibaba and Soochow University.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of ACL*, pages 2442–2452.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of EMNLP 2012*, pages 1455–1465.

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theory*, pages 24–41.

Wanxiang Che, Zhenghua Li, and Ting Liu. 2012. Chinese Dependency Treebank 1.0 (LDC2012T05). In *Philadelphia: Linguistic Data Consortium*.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of EMNLP*, pages 740–750.

Keh-Jiann Chen, Chi-Ching Luo, Ming-Chung Chang, Feng-Yi Chen, Chao-Jan Chen, Chu-Ren Huang, and Zhao-Ming Gao. 2003. *Sinica treebank: Design criteria,representational issues and implementation*, chapter 13. Kluwer Academic Publishers.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of ICML*.

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

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependecy parsing. In *Proceedings of ICLR*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of ACL*, pages 334–343.

Jan Einarsson. 1976. Talbankens skriftspråkskonkordans. Department of Scandinavian Languages, Lund University.

Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2016. A universal framework for inductive transfer parsing across multi-typed treebanks. In *Proceedings of COLING*, pages 12–22.

Lijuan Guo, Zhenghua Li, Xue Peng, and Min Zhang. 2018. Data annotation guideline of chinese dependency syntax for multi-domain and multi-source texts. *Journal of Chinese Information Processing*.

Jerker Järborg. 1986. Manual for syntaggning. Department of Linguistic Computation, University of Gothenburg.

Richard Johansson. 2013. Training parsers on incompatible treebanks. In *Proceedings of NAACL*, pages 127–137.

Xiang Li, Wenbin Jiang, Yajuan Lü, and Qun Liu. 2013. Iterative transformation of annotation guidelines for constituency parsing. In *Proceedings of ACL*, pages 591–596.

Zhenghua Li, Wanxiang Che, and Ting Liu. 2012. Exploiting multiple treebanks for parsing with quasisynchronous grammar. In *Proceedings of ACL*, pages 675–684.

Zhenghua Li, Min Zhang, Yue Zhang, Zhanyi Liu, Wenliang Chen, Hua Wu, and Haifeng Wang. 2016. Active learning for dependency parsing with partial annotation. In *Proceedings of ACL*.

Xuezhe Ma and Eduard Hovy. 2017. Neural probabilistic model for non-projective mst parsing. In *Proceedings of IJCNLP*, pages 59–69.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *Proceedings of ACL*, pages 1105–1116.

Simonetta Montemagni, Francesco Barsotti, Marco Battista, Nicoletta Calzolari, Ornella Corazzari, Alessandro Lenci, Antonio Zampolli, Francesca Fanciulli, Maria Massetani, Remo Raffaelli, Roberto Basili, Maria Teresa Pazienza, Dario Saracino, Fabio Zanzotto, Nadia Mana, Fabio Pianesi, and Rodolfo Delmonte. 2003. Building the italian syntactic–semantic treebank. In Anne Abeille, editor, Building and Using Syntactically Annotated Corpora. Kluwer, Dordrecht.

Zheng-Yu Niu, Haifeng Wang, and Hua Wu. 2009. Exploiting heterogeneous treebanks for parsing. In *Proceedings of ACL*, pages 46–54.

Eric W. Noreen. 1989. *Computer-intensive methods for testing hypotheses: An introduction*. John Wiley & Sons, Inc., New York.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543.

Likun Qiu, Yue Zhang, Peng Jin, and Houfeng Wang. 2014. Multi-view chinese treebanking. In *Proceedings of COLING*, pages 257–268.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of ACL*, pages 1556–1566.

Heike Telljohann, Erhard Hinrichs, and Sandra Kbler. 2004. The Tüba-D/Z treebank: Annotating German with a context-free backbone. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC)*, pages 2229–2235.

Nianwen Xue, Fei Xia, Fu-Dong Chiou, and Martha Palmer. 2005. The Penn Chinese Treebank: Phrase structure annotation of a large corpus. In *Natural Language Engineering*, volume 11, pages 207–238.

Weidong Zhan. 2012. The application of treebank to assist Chinese grammar instruction: a preliminary investigation. *Journal of Technology and Chinese Language Teaching*, 3(2):16–29.

Meishan Zhang, Jie Yang, Zhiyang Teng, and Yue Zhang. 2016. Libn3l:a lightweight package for neural nlp. In *Proceedings of LREC*, pages 225–229.

Hao Zhou, Yue Zhang, Shujian Huang, and Jiajun Chen. 2015. A neural probabilistic structured-prediction model for transition-based dependency parsing. In *Proceedings of ACL*, pages 1213–1222.

Qiang Zhou. 2004. Annotation scheme for Chinese treebank. *Journal of Chinese Information Processing*, 18(4):1–8.

Muhua Zhu, Jingbo Zhu, and Minghan Hu. 2011. Better automatic treebank conversion using a feature-based approach. In *Proceedings of ACL*, pages 715–719.