Mimic and Conquer: Heterogeneous Tree Structure Distillation for Syntactic NLP

Hao Fei[1], **Yafeng Ren**[2] and **Donghong Ji**[1*]
1. Department of Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, China
2. Guangdong University of Foreign Studies, China
{hao.fei,renyafeng,dhji}@whu.edu.cn

Abstract

Syntax has been shown useful for various NLP tasks, while existing work mostly encodes singleton syntactic tree using one hierarchical neural network. In this paper, we investigate a simple and effective method, Knowledge Distillation, to integrate heterogeneous structure knowledge into a unified sequential LSTM encoder. Experimental results on four typical syntax-dependent tasks show that our method outperforms tree encoders by effectively integrating rich heterogeneous structure syntax, meanwhile reducing error propagation, and also outperforms ensemble methods, in terms of both the efficiency and accuracy.

1 Introduction

Integrating syntactic information into neural networks has received increasing attention in natural language processing (NLP), which has been used for a wide range of end tasks, such as sentiment analysis (SA) (Nguyen and Shirai, 2015; Teng and Zhang, 2017; Looks et al., 2017; Zhang and Zhang, 2019), neural machine translation (NMT) (Cho et al., 2014; Garmash and Monz, 2015; Gū et al., 2018), language modeling (Yazdani and Henderson, 2015; Zhang et al., 2016; Zhou et al., 2017), semantic role labeling (SRL) (Marcheggiani and Titov, 2017; Strubell et al., 2018; Fei et al., 2020c), natural language inference (NLI) (Tai et al., 2015a; Liu et al., 2018) and text classification (Chen et al., 2015; Zhang et al., 2018b). Despite the usefulness of structure knowledge, most existing models use only a single syntactic tree, such as a constituency or a dependency tree.

Constituent and dependency representation for syntactic structure share underlying linguistic and computational characteristics, while differ also in various aspects. For example, the former focuses

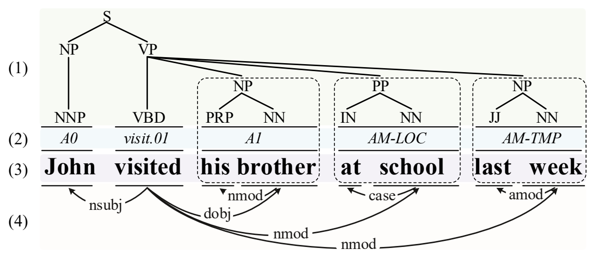

Figure 1: An example illustrating the mutual benefit of constituency and dependency tree structures. (1) refers to the constituency tree structure, (2) indicates the semantic role labels, (3) refers to the example sentence, (4) represents the dependency tree structure.

on revealing the continuity of phrases, while the latter is more effective in representing the dependencies among elements. By integrating the two representations from heterogeneous trees, the mutual benefit has been explored for joint parsing tasks (Collins, 1997; Charniak and Johnson, 2005; Farkas et al., 2011; Yoshikawa et al., 2017; Zhou and Zhao, 2019). Intuitively, complementary advantages from heterogeneous trees can facilitate a range of NLP tasks, especially syntax-dependent ones such as SRL and NLI. Taking the sentence of Figure 1 as example, where an example is shown from SRL[1] task. In this case, the dependency links can locate the relations between arguments and predicates more efficiently, while the constituency structure can aggregate the phrasal spans for arguments, and guide the global path to the predicate. Integrating the features of two structures can better guide the model to focus on the most suitable phrasal granularity (as circled by the dotted box), and also ensure the route consistency between the semantic objective pairs.

In this paper, we investigate the *Knowledge Distillation* (KD) method, which has been shown to be

[*]Corresponding author.

[1]We consider the span-based SRL, which aims to annotate the phrasal span of all semantic arguments.

effective for knowledge ensembling (Hinton et al., 2015; Kim and Rush, 2016; Furlanello et al., 2018), for heterogeneous structure integration. Specifically, we employ a sequential LSTM as the student for distilling heterogeneous syntactic structures from various teacher tree encoders, such as GCN (Kipf and Welling, 2017) and TreeLSTM (Tai et al., 2015a). We consider output distillation, syntactic feature injection and semantic learning. In addition, we introduce an alternative structure injection strategy to enhance the ability of heterogeneous syntactic representations within the shared sequential model. The distilled structure-aware student model can make inference using sequential word inputs alone, reducing the error accumulation from external parsing tree annotations.

We conduct extensive experiments on a wide range of syntax-dependent tasks, including semantic role labeling, relation classification, natural language inference and sentiment classification. Results show that the distilled student outperforms tree encoders, verifying the advantage of integrating heterogeneous structures. The proposed method also outperforms existing ensemble methods and strong baseline systems, demonstrating its high effectiveness on structure information integration.

2 Related Work

2.1 Syntactic Structures for Text Modeling

Previous work shows that integrating syntactic structure knowledge can improve the performance of NLP tasks (Socher et al., 2013; Cho et al., 2014; Nguyen and Shirai, 2015; Looks et al., 2017; Liu et al., 2018; Zhang and Zhang, 2019; Fei et al., 2020b). Generally, these methods consider injecting either standalone constituency tree or dependency tree by tree encoders such as TreeLSTM (Socher et al., 2013; Tai et al., 2015a) or GCN (Kipf and Welling, 2017). Based on the assumption that the dependency and constituency representation can be disentangled and coexist in one shared model, existing efforts are paid for joint constituent and dependency parsing, verifying the mutual benefit of these heterogeneous structures (Collins, 1997; Charniak, 2000; Charniak and Johnson, 2005; Farkas et al., 2011; Ren et al., 2013; Yoshikawa et al., 2017; Strzyz et al., 2019; Kato and Matsubara, 2019; Zhou and Zhao, 2019). However, little attention is paid for facilitating the syntax-dependent tasks via integrating heterogeneous syntactic trees. Although the integration from heterogeneous trees can be achieved via widely employed approaches, such as ensemble learning (Wolpert, 1992; Ju et al., 2019) and multi-task training (Liu et al., 2016; Chen et al., 2018; Fei et al., 2020a), they usually suffer from low-efficiency and high computational complexity.

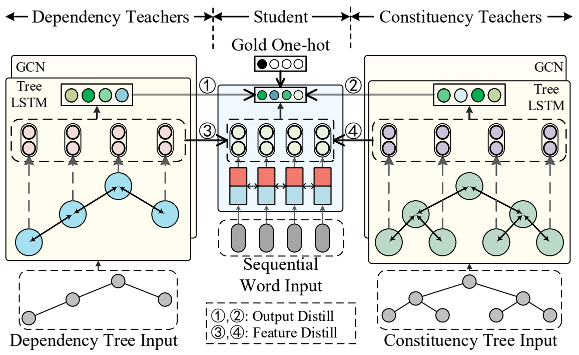

Figure 2: Overall framework of the proposed model.

2.2 Knowledge Distillation

Our work is related to knowledge distillation techniques. It has been shown that KD is very effective and scalable for knowledge ensembling (Hinton et al., 2015; Furlanello et al., 2018), and existing methods are divided into two categories: 1) *output distillation*, which makes a teacher model output logits as a student model training objective (Kim and Rush, 2016; Vyas and Carpuat, 2019; Clark et al., 2019), 2) *feature distillation*, which allows a student to learn from a teacher's intermediate feature representations (Zagoruyko and Komodakis, 2017; Sun et al., 2019). In this paper, we enhance the distillation of heterogeneous structures via both output and feature distillations by employing a sequential LSTM as the student. Our work is also closely related to Kuncoro et al., (2019), who distill syntactic structure knowledge to a student LSTM model. The difference lies in that they focus on transferring tree knowledge from syntax-aware language model for achieving scalable unsupervised syntax induction, while we aim at integrating heterogeneous syntax for improving downstream tasks.

3 Method

As shown in Figure 2, the overall architecture consists of a sequential LSTM (Hochreiter and Schmidhuber, 1997) student, and several tree teachers for dependency and constituency structures.

3.1 Tree Encoder Teachers

Different tree models can encode the same tree structure, resulting in different heterogeneous tree representations. Following previous work (Tai et al., 2015b; Marcheggiani and Titov, 2017; Zhang and Zhang, 2019), we consider encoding dependency trees by Child-Sum TreeLSTM and constituency trees by N-ary TreeLSTM. We also employ GCN to encode dependency and constituency structures separately. We employ a bidirectional tree encoder to fully capture the structural information interaction. Formally, we denote $\boldsymbol{X} = \{x_1, \cdots, x_n\}$ as an input sentence, $\boldsymbol{X}^{\text{dep}} = \{x_1^{\text{dep}}, \cdots, x_n^{\text{dep}}\}$ as the dependency tree and $\boldsymbol{X}^{\text{con}} = \{x_1^{\text{con}}, \cdots, x_n^{\text{con}}\}$ as the constituency tree.

Encoding dependency structure. We first use the standard Child-Sum TreeLSTM to encode the dependency structure, where each node j in the tree takes as input the embedding vector x_j corresponding to the head word. The conventional bottom-up fashion is:

$$\begin{aligned}
\overline{h}_j &= \sum_{k \in C(j)} h_k \\
i_j &= \sigma(\boldsymbol{W}^{(i)} x_j^{\text{dep}} + \boldsymbol{U}^{(i)} \overline{h}_j + b^{(i)}) \\
f_{jk} &= \sigma(\boldsymbol{W}^{(f)} x_j^{\text{dep}} + \boldsymbol{U}^{(f)} \overline{h}_k + b^{(f)}) \\
o_j &= \sigma(\boldsymbol{W}^{(o)} x_j^{\text{dep}} + \boldsymbol{U}^{(o)} \overline{h}_j + b^{(o)}) \\
u_j &= \tanh(\boldsymbol{W}^{(u)} x_j^{\text{dep}} + \boldsymbol{U}^{(u)} \overline{h}_j + b^{(u)}) \\
c_j &= i_j \odot u_j + \sum_{k \in C(j)} f_{jk} \odot c_k \\
h_j &= o_j \odot \tanh(c_j)
\end{aligned} \tag{1}$$

where $\boldsymbol{W}$, $\boldsymbol{U}$ and b are parameters. $C(j)$ refers to the set of child nodes of j. h_j, i_j, o_j and c_j are the hidden state, input gate, output gate and memory cell of the node j, respectively. f_{jk} is a forget gate for each child k of j. $\sigma(\cdot)$ is an activation function and $\odot$ is element-wise multiplication. Similarly, the top-down TreeLSTM has the same transition equations as the bottom-up TreeLSTM, except that the direction and the number of dependent nodes are different. We concatenate the tree representations of two directions for each node: $h_j^{bi} = [h_j^{\uparrow}; h_j^{\downarrow}]$.

Compared with TreeLSTM, GCN is more computationally efficient in performing the tree propagation for each node in parallel with O(1) complexity. Considering the constructed dependency graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ are sets of nodes, and $\mathcal{E}$ are sets of bidirectional edges between heads and dependents, respectively. GCN can be viewed as a hierarchical node encoder, representing the node j at the l-th layer encoded as follows:

$$g_i^l = \sigma(\boldsymbol{W}_i^l h_i^l + b_i^l) \tag{2}$$

$$h_j^l = \text{ReLU}(\textstyle\sum_{i \in \mathcal{N}(i)} x_i^l \odot g_i^l) \tag{3}$$

where $\mathcal{N}(i)$ are neighbors of the node j. ReLU is a non-linear activation function. For dependency encoding by TreeLSTM or GCN, we make use of all the node representations, $\boldsymbol{R}^{\text{dep}} = [r_1, \cdots, r_n]$, within the whole tree structure for next distillation.

Encoding constituency structure. We employ N-ary TreeLSTM to encode constituent tree:

$$\begin{aligned}
i_j &= \sigma(\boldsymbol{W}^{(i)} x_j^{\text{con}} + \sum_{q=1}^{N} \boldsymbol{U}_q^{(i)} h_{jq} + b^{(i)}) \\
f_{jk} &= \sigma(\boldsymbol{W}^{(f)} x_j^{\text{con}} + \sum_{q=1}^{N} \boldsymbol{U}_{kq}^{(f)} h_{jq} + b^{(f)}) \\
o_j &= \sigma(\boldsymbol{W}^{(o)} x_j^{\text{con}} + \sum_{q=1}^{N} \boldsymbol{U}_q^{(o)} h_{jq} + b^{(o)}) \\
u_j &= \tanh(\boldsymbol{W}^{(u)} x_j^{\text{con}} + \sum_{q=1}^{N} \boldsymbol{U}_q^{(u)} h_{jq} + b^{(u)}) \\
c_j &= i_j \odot u_j + \sum_{q=1}^{N} f_{jq} \odot c_{jq} \\
h_j &= o_j \odot \tanh(c_j)
\end{aligned} \tag{4}$$

where q is the index of the branch of j. Slightly different from Child-Sum TreeLSTM, the separate parameter matrices for each child k allow the model to learn more fine-grained and order-sensitive children information. We also concatenate two directions from both bottom-up and top-down of each node as the final representation.

Similarly, GCN is also used to encode the constituent graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ via Eq. (2) and (3). Note that there are both words and constituent labels in the node set $\mathcal{V}$. For constituency encoding by both TreeLSTM and GCN, we take the representations of terminal nodes in the structure as the corresponding word representations $\boldsymbol{R}^{\text{con}} = [r_1, \cdots, r_n]$.

3.2 Heterogeneous Structure Distillation

Sequential models have been proven effective on encoding syntactic tree information (Shen et al., 2018; Kuncoro et al., 2019). We set the goal of KD as simultaneously distilling heterogeneous structures from tree encoder teachers into a LSTM student model.

We denote $\Gamma(\text{dep}) = \{\gamma(\text{TreeLSTM}), \gamma(\text{GCN})\}$ as the dependency teachers, and $\Gamma(\text{con}) = \{\gamma(\text{TreeLSTM}), \gamma(\text{GCN})\}$ as the constituency teachers, and $\Gamma(\text{all})=\Gamma(\text{dep}) \bigcup \Gamma(\text{con})$ as the overall teachers. The objective of the student model can be decomposed into three terms: an output distillation target, a semantic target, and a syntactic target.

Output distillation. The output logits serve as *soft target* providing richer supervision than the *hard target* of one-hot gold label for the training (Hinton et al., 2015). Given an input sentence $\boldsymbol{X}$ with the gold label Y (one-hot), the output logits of teachers are $P^t_{\Gamma(\text{all})}$, and the output logits of the student is P^s. The output distilling can be denoted as:

$$\mathcal{L}_{\text{output}} = \mathcal{H}([\alpha Y + (1-\alpha) P^t_{\Gamma(\text{all})}], P^s) \tag{5}$$

where $\mathcal{H}(,)$ refers to the cross-entropy. α is a coupling factor, which increases from 0 to 1 in training, namely *teacher annealing* (Clark et al., 2019).

Syntactic tree feature distillation. In order to capture rich syntactic tree features, we consider allowing the student to directly learn from the teachers' feature hidden representation. Specifically, we denote the hidden representation of the student LSTM as $\boldsymbol{R}^{\text{s}} = [r_1, \cdots, r_n]$, and we expect $\boldsymbol{R}^{\text{s}}$ to be able to predict the output of $\boldsymbol{R}^{\text{dep}}$ or $\boldsymbol{R}^{\text{con}}$ from syntax-aware teachers. Thus the target is to optimize the following regression loss:

$$\mathcal{L}^{(A)}_{\text{dep}} = \frac{1}{2} \sum_{j=1}^{n} || f^t_{\Gamma(\text{dep})}(r^{\text{dep}}_j) - f^s(r^{\text{s}}_j) ||^2 \tag{6}$$

$$\mathcal{L}^{(A)}_{\text{con}} = \frac{1}{2} \sum_{j=1}^{n} || f^t_{\Gamma(\text{con})}(r^{\text{con}}_j) - f^s(r^{\text{s}}_j) ||^2 \tag{7}$$

$$\mathcal{L}^{(A)}_{\text{syn}} = \eta \mathcal{L}^{(A)}_{\text{dep}} + (1-\eta) \mathcal{L}^{(A)}_{\text{con}} \tag{8}$$

where $\eta \in [0, 1]$ is a factor for coordinating the dependency and constituency structure encoding, $f^t_{\Gamma(\text{dep})}()$, $f^t_{\Gamma(\text{con})}()$, $f^s()$ are the feedforward layers, respectively, for calculating the corresponding score vectors, and j is the word index.

Semantic learning. We randomly mask a target input word Q_j and let LSTM predict the word based on its hidden representation of prior words. In consequence, we pose the following language modeling objective:

$$\mathcal{L}_{\text{sem}} = \sum_{j=1}^{M} \mathcal{H}(Q_j, P^s_j | \boldsymbol{X}_{[1, \cdots, j-1]}) \tag{9}$$

by which LSTM can additionally improve the ability of semantic learning.

3.3 Enhanced Structure Injection

We consider further enhancing the trees injection, by encouraging the student to mimic the dependency and constituency tree induction of teachers.

Dependency injection. We force the student to predict the distributions of dependency arcs and labels based on the hidden representations and the representations of teachers.

$$\mathcal{L}^{(B)}_{\text{dep}} = \sum_{j}^{n} \sum_{i}^{n} \mathcal{H}(P^t_{\Gamma(\text{dep})}(r_j|x_i), P^s(r_j|x_i)) + \sum_{j}^{n} \sum_{i}^{n} \sum_{k}^{L} \mathcal{H}(P^t_{\Gamma(\text{dep})}(l_k|r_j, x_i), P^s(l_k|r_j, x_i)) \tag{10}$$

where $P^t_{\Gamma(\text{dep})}(r_j|x_i)$ is the arc probability of the parent node r_j for x_i in the dependency teacher, and $P^t_{\Gamma(\text{dep})}(l_k|r_j, x_i)$ is the probability of the label l_k for the arc (r_j, x_i) in the teacher.

Constituency injection. Similarly, to enhance constituency injection, we mimic the distribution of each span (i, j) with the label l in teachers. Following Zhou et al. (2019), we adopt a feedforward layer as the span scorer:

$$\text{Scr}(t) = \sum_{(i,j)\in t} \sum_{k} f(i, j, l) \tag{11}$$

We use the CYK algorithm (Cocke, 1970; Younger, 1975; Kasami, 1965) to search the highest score tree T^* in teachers, and all possible trees T in the student. Then we optimize the following hinge loss between the structures in the student and teachers:

$$\mathcal{L}^{(B)}_{\text{con}} = \max(0, \max_{t\in T}(\text{Scr}(t) + \Delta(t, T^*)) - \text{Scr}(T^*)) \tag{12}$$

where Δ is the hamming distance. The above syntax loss in Eq. (10) and (12) can substitute the ones in Eq. (6) and (7), respectively. The overall objective of the structure injection is:

$$\mathcal{L}^{(B)}_{\text{syn}} = \eta \mathcal{L}^{(B)}_{\text{dep}} + (1-\eta) \mathcal{L}^{(B)}_{\text{con}} \tag{13}$$

Regularization. Based on the independent assumption, the syntax feature distillations target learning diversified private representations for heterogeneous structures as much as possible. In practice, there should be a latent shared structure in the parameter space, while the separate distillations will squeeze such shared feature, weakening the expression of the learnt representations. To avoid this, we additionally impose a regularization on Eq. (6), (7), (10) and (12):

$$\mathcal{L}_{\text{reg}} = \frac{\zeta}{2}||\Theta||^2 , \tag{14}$$

where Θ is the overall parameter in the student.

3.4 Training

Algorithm 1 gives the overall structure distillation process. At early training stage (line 2-19), semantic learning (Eq. 9) and output distillation (Eq. 5) are first executed by each teacher. As we have multiple teachers for one student on each task, for syntactic tree structure distillation, we sequentially distill one teacher at one time. We take turn with a turning gap G_2 processing the dependency or constituency injection from a tree teacher (line 13-17), to keep the training stable. After a certain number of training iterations G_1, we optimize the overall loss (line 20):

$$\mathcal{L}_{\text{all}} = \mathcal{L}_{\text{output}} + \lambda_1 \mathcal{L}_{\text{syn}} + \lambda_2 \mathcal{L}_{\text{sem}} \tag{15}$$

where λ_1 and λ_2 are coefficients, which regulate the corresponding objectives. $\mathcal{L}_{\text{syn}}$ can be either $\mathcal{L}^{(A)}_{\text{syn}}$ (Eq. 8) or $\mathcal{L}^{(B)}_{\text{syn}}$ (Eq. 13), that is, the syntax sources are simultaneously from two tree encoders (dependency and constituency) at one time. During inference, the well-trained student can make prediction alone with only word sequential input.

Algorithm 1: Distill heterogeneous trees.

```
   Input: Training set: $(\boldsymbol{X}, \boldsymbol{X}^{\text{dep}}, \boldsymbol{X}^{\text{con}}, Y)$;
          Total iteration $T$; Syntax turn gaps
          $G_1$, $G_2$; Syntax flag $F$=Ture.
   Output: Student model.
 1 while $t < T$ do
 2     if $t \leq G_1$ then
 3         if $t\%G_2$ == 0 then
 4             $F \leftarrow !F$ ;
 5         end
 6         $P^s \leftarrow$ Student($\boldsymbol{X}$) ;
 7         opt $\mathcal{L}_{\text{sem}}$ in Eq. (9) ;
 8         for $\gamma(\textit{model}) \in \Gamma(\textit{all})$ do
 9             $P^t_{\Gamma(\text{dep})}, r^{\text{dep}}_j \leftarrow \gamma(\text{model})(\boldsymbol{X}^{\text{dep}})$;
10             $P^t_{\Gamma(\text{con})}, r^{\text{con}}_j \leftarrow \gamma(\text{model})(\boldsymbol{X}^{\text{con}})$;
11             $P^t_{\Gamma(\text{all})} = P^t_{\Gamma(\text{dep})} \bigcup P^t_{\Gamma(\text{con})}$ ;
12             opt $\mathcal{L}_{\text{output}}$ in Eq. (5) ;
13             if $F$ then
14                 opt $\mathcal{L}_{\text{dep}}$ in Eq. (6) or (10) ;
                   // dependency learning
15             else
16                 opt $\mathcal{L}_{\text{con}}$ in Eq. (7) or (12) ;
                   // constituency learning
17             end
18         end
19     else
20         opt $\mathcal{L}_{\text{all}}$ in Eq. (15) ;
21     end
22 end
```

4 Experiments

4.1 Experimental Setups

Hyperparameters. We use a 3-layer BiLSTM as our student, and a 2-layer architecture for all tree teachers. The default word embeddings are initialized randomly, and its dimension is set as 300. The hidden size is set to 350 in the student LSTM, and 300 in the teacher models, respectively. We adopt the Adam optimizer with an initial learning rate of 1e-5. We use the mini-batch of 32 within total 10k (T) iterations with early stopping, and apply 0.4 dropout ratio for all embeddings. We set the coefficients λ_1, λ_2, ζ and η as 0.6, 0.2, 0.2 and 0.5, respectively. The training iteration thresholds G_1 and G_2 are set as 300 and 128, respectively. These values achieve the best performance in the development experiments.

Baselines systems. We compare the following baselines. 1) Sequential encoders: LSTM, attention-based LSTM (AttLSTM) and Transformer (Vaswani et al., 2017), sentence-state LSTM (S-LSTM) (Zhang et al., 2018a); 2) Tree encoders introduced in §2; 3) Ensemble models: ensembling learning (EnSem) (Wolpert, 1992; Ju et al., 2019), multi-task method (MTL) (Liu et al., 2016; Chen et al., 2018), adversarial training (AdvT) (Liu et al., 2017) and tree communication model (TCM) (Zhang and Zhang, 2019). For EnSem, we only concatenate the output representations of tree encodes. For MTL, we use an underlying shared LSTM for parameter sharing for tree encodes. For

AdvT, we adopt the shared-private architecture (Liu et al., 2017) based on MTL. Following Zhang et al. (2019), for TCM, we initialize GCN for TreeLSTM, to encode dependency and constituency trees respectively, and finally concatenate the output representations. Note that all the models in Tree Ensemble group take total four Tree teachers as in our distillation teachers' meta-encoders. 4) Other baselines: ESIM (Chen et al., 2017), local-global pattern based self-attention networks (LG-SANs) (Xu et al., 2019) and BERT.

Tasks and evaluation. The experiments are conducted on four representative syntax-dependent tasks: 1) `Rel`, relation classification on Semeval10 (Hendrickx et al., 2010); 2) `NLI`, sentence pair classification on the Stanford NLI (Bowman et al., 2015); 3) `SST`, binary sentiment classification task on the Stanford Sentiment Treebank (Socher et al., 2013), 4) `SRL`, semantic role labeling on the CoNLL2012 OntoNotes (Pradhan et al., 2013). For `NLI`, we make element-wise production, subtraction, addition and concatenation of two separate sentence representations as a whole. We mainly adopt F1 score to evaluate the performance of different models. The data splitting follows previous work.

Trees annotations and resources. The OntoNotes data offers the annotations of the dependency and constituency structure. For the rest datasets, we parse sentences via the state-of-the-art BiAffine dependency parser (Dozat and Manning, 2017), and the Self-Attentive constituency parser (Kitaev and Klein, 2018). The parsers are trained on PTB[2]. The dependency parser has a 93.4% LAS, and the constituency parser has 92.6% F1 score. Besides, we evaluate different contextualized word representations, such as ELMo[3] and BERT[4].

4.2 Main Results

Experimental results of different models are shown in Table 1, where several observations can be found. First, tree models encoded with syntactic knowledge can facilitate syntax-dependent tasks, outperforming sequential models by a substantial margin. Second, different tree encoders integrated with varying syntactic tree structures can make different contributions to the tasks. For example, GCN with dependency structure gives the best result for `Rel`, while TreeLSTM with constituency tree achieves the best performance for `SST`. Third, when integrating heterogeneous tree structures by tree ensemble methods, a competitive performance can be obtained, showing the importance of integrating heterogeneous tree information. Finally, our distilled student model significantly outperforms all the baseline systems[5], demonstrating its high effectiveness on the integration of heterogeneous structure information.

	Rel	**NLI**	**SST**	**SRL**
• **Sequential Encoder**				
LSTM	80.5	79.6	82.3	76.6
AttLSTM	82.3	81.5	84.2	78.2
Transformer	84.7	84.2	85.0	80.5
S-LSTM	85.0	84.8	86.2	82.0
• **Standalone Tree Model**				
TreeLSTM+dep.	85.2	86.0	86.4	82.5
GCN+dep.	85.9	85.8	86.1	83.3
TreeLSTM+con.	85.0	86.8	87.6	82.2
GCN+con.	84.8	86.3	86.8	81.8
Avg.	85.3	86.2	86.5	82.4
• **Tree Ensemble**				
EnSem	85.5	87.0	86.0	81.4
MTL	84.9	88.3	87.2	83.7
AdvT	86.4	87.6	85.2	82.1
TCM	85.7	88.8	88.4	83.0
Avg.	85.9	88.1	86.7	82.3
• **Distilled Student**				
`Best`	**89.2***	**90.8***	**91.6***	**85.5***
• **Others**				
ESIM	-	88.9	-	-
LG-SANs	85.6	86.5	87.3	81.2
BERT	<u>91.3</u>	<u>92.1</u>	<u>94.4</u>	<u>86.0</u>

Table 1: Main results on various end tasks. $*$ indicates $p \leq$0.05.

Ablation results. We ablate each part of our distilling method in Table 2. First, we find that the enhanced structure injection strategy ($\mathcal{L}_{syn}^{(B)}$) can help to achieve the best results for all the tasks, compared with the latent syntax feature mimic ($\mathcal{L}_{syn}^{(A)}$). By ablating each distilling objective, we learn that the syntax tree distillation ($\mathcal{L}_{syn}$) is the kernel of our knowledge distillation for these

[2]https://catalog.ldc.upenn.edu/LDC99T42.

[3]https://allennlp.org/elmo

[4]https://github.com/google-research/bert

[5]Note that a direct comparison with BERT is unfair, because a large number of pre-trained parameters can bring overwhelming improvement.

	Rel	NLI	SST	SRL
• Syntax Injection Strategy				
$+\mathcal{L}_{syn}^{(A)}$	88.6	90.2	91.0	85.0
$+\mathcal{L}_{syn}^{(B)}$	89.2	90.8	91.6	85.5
• Distilling Objective (with $\mathcal{L}_{syn}^{(B)}$)				
w/o $\mathcal{L}_{sem}$	87.9	89.2	89.7	84.8
w/o $\mathcal{L}_{syn}$	86.8	88.7	88.9	83.7
w/o $\mathcal{L}_{reg}$	88.1	89.3	89.9	84.7
w/o Tea.Anl.	88.2	89.1	90.4	84.5
• Contextualized Semantics (with $\mathcal{L}_{syn}^{(B)}$)				
+ELMo	90.6	91.6	92.4	85.1
+BERT	**92.2**	**93.0**	**95.1**	**86.8**

Table 2: Ablation results on distilled student. 'Tea.Anl.' refers to teacher annealing. In 'Semantics', we replace semantic learning $\mathcal{L}_{sem}$ with pre-trained contextualized word representations.

	Constituency	Dependency
TreeLSTM+dep.	28.31	73.92
GCN+dep.	19.11	**76.32**
TreeLSTM+con.	**68.65**	30.27
GCN+con.	66.30	23.85
Student-Full	62.61	70.34
w/o $\mathcal{L}_{reg}$	53.20	64.08

Table 3: Probing the upper-bound of constituent and dependent syntactic structure.

syntax-dependent tasks, compared with semantic feature learning ($\mathcal{L}_{sem}$). Besides, both the introduced teacher annealing factor α and regularization $\mathcal{L}_{reg}$ can benefit the task performance. Finally, we explore recent contextualized word representations, including ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019). Surprisingly, our distilled student model receives a substantial performance improvements in all tasks. However, when removing the proposed syntax distillation from BERT, the performance drops, as shown in Table 1 (the vanilla BERT).

4.3 Heterogeneous Tree Structure

Upper-bound of heterogeneous structures. We explore to what extent the distilled student can manage to capture heterogeneous tree structure information. Following previous work (Conneau et al., 2018), we consider employing two syntactic probing tasks, including 1) **Constituent labeling**, which assigns a non-terminal label for text spans within the phrase-structure (e.g., *Verb*, *Noun*, etc.), and 2) **Dependency labeling**, which predicts the

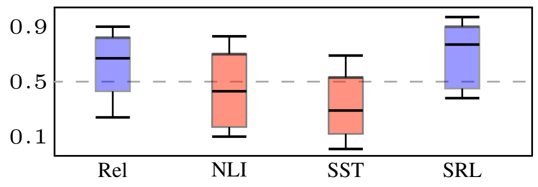

Figure 3: Heterogeneous syntax distribution. The predominance of dependency syntax is above 0.5, otherwise for constituency.

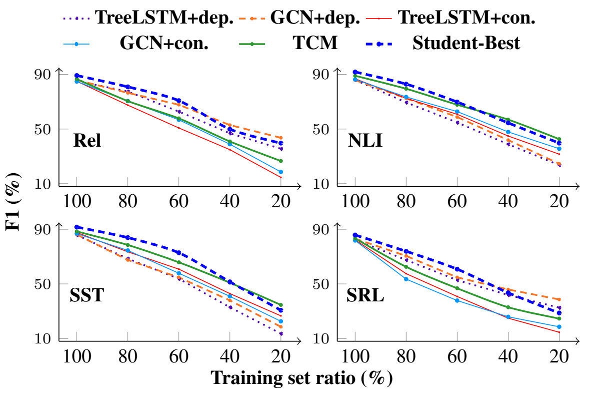

Figure 4: Results under varying ratio of train set.

relationship (edge) between two tokens (e.g., *subject-object* etc.). We take the last-layer output representation as the probing objective. We compare the student model with four teacher tree encoders, separately, based on the SRL task. As shown in Table 3, the student LSTM gives slightly lower score than one of the best tree models (i.e., *GCN+dep.* for dependency labeling, *TreeLSTM+con.* for constituency labeling), showing the effectiveness on capturing syntax. Besides, we can find that the regularization $\mathcal{L}_{reg}$ plays a key role in improving the expression capability of the learnt representation.

Distributions of heterogeneous syntax in different tasks. We also compare the distributions of dependency and constituency structures in different tasks after fine-tuning. Technically, based on each example in the test set, the performance drops when the student LSTM is trained only under either standalone dependency or constituency injection (TreeLSTM or GCN), respectively, by controlling η=0 or 1. Intuitively, the more the results drop, the more the model benefits from the corresponding syntax. For each task, we collect the sensitivity values and linearly normalize them into [0,1] for all examples, and make statistics. As plotted in Figure 3, the distributions of dependency and constituency syntax vary among tasks, verifying that different tasks depend on distinct types of structural knowledge, while integrating them altogether can give the best effects. For example, `TreeDepth`, dependency structures support `Rel` and `SRL`, while `NLI` and `SST` benefit from constituency the most.

4.4 Robustness Analysis

Generalization ability to training data. Figure 4 shows the performance of different models on varying ratio of the full training dataset. We can find that the performance decreases with the reduction of the training data for all methods, while our distilled student achieves better results, compared with most of the baselines. The underlying reasons are two-fold. First, the heterogeneous syntactic features can provide strong representations for supporting better predictions. Second, the distilled student takes only sequential inputs, avoiding the noise from parsed inputs to some extent.

Also we see that *TreeLSTM/GCN+dep.* can counteract the data reduction ($\leq$40%) on `Rel` and `SRL` tasks, showning that they rely more on dependency structures, while `NLI` and `SST` depend on constituency structures. In addition, the student starts underperforming than the best one on the small data ($\leq$40%). Without explicit tree annotations, the contribution of heterogeneous syntax can be deteriorated. But it still remains robust on shortage of training data than most of the baselines, due to its noise resistant.

Reducing error accumulation of tree annotation. We investigate the effects on reducing noises from tree annotation. We compare the performance under different sources. Table 4 shows the results on `SRL`. With only word inputs, our model still outperforms the baselines which take the gold syntax annotation. This partially shows that without parsed tree annotation, the student model can avoid noise and error propagation. When we add gold annotation as additional signal, the performance can be further improved.

Efficiency study. As shown in Figure 5, the student model has fewer parameters, while keeping faster decoding speed, compared with other ensemble models. Our sequential model is about 3 times smaller than AdvT, but nearly 4 times faster than the tree ensemble methods. Such observation coincides with previous studies (Kim and Rush, 2016; Sun et al., 2019; Clark et al., 2019).

System	Auto-Syn	Gold-Syn	w/o Syn
TreeLSTM+dep.	80.6	82.5	-
GCN+dep.	81.1	83.3	-
TreeLSTM+con.	79.6	82.2	-
GCN+con.	79.8	81.8	-
EnSem	80.5	81.4	-
MTL	81.2	83.7	-
AdvT	81.0	82.1	-
TCM	82.4	83.0	-
Student-Full	-	**86.2**†	**85.5**

Table 4: Performance of different systems with automatically-parsed/gold syntax, and without syntax annotations. † indicates that we concatenate additional gold syntactic label with other input features.

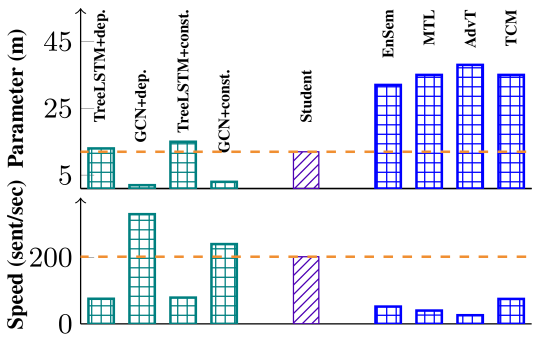

Figure 5: Comparisons on parameter scale and decoding speed.

4.5 Visualization on Heterogeneous Structure

The enhanced structure injection objectives (Eq. (10) and (12)) enables the student LSTM to unsupervisedly induce tree structures at the test stage. To understand how the distilled model promote the mutual learning of heterogeneous structures, we empirically visualize the induced trees based on a test example of SRL. As shown in Figure 6, the discovered dependency structures accurately match the gold tree, and the constituents are highly correlated with the gold one. Besides, the edges that indicate the two elements are augmented by the learning of each other, which in return enhance the recognition of the spans of elements (yellow dotted boxes), respectively. For example, the constituent and dependent paths (green lines) linking two minimal target spans, *the Focus Today program* and *by Wang Shilin*, are enhanced and echoed with each other, via the core predicate. This reveals that our method can offer a deeper latent interaction between heterogeneous tree structures.

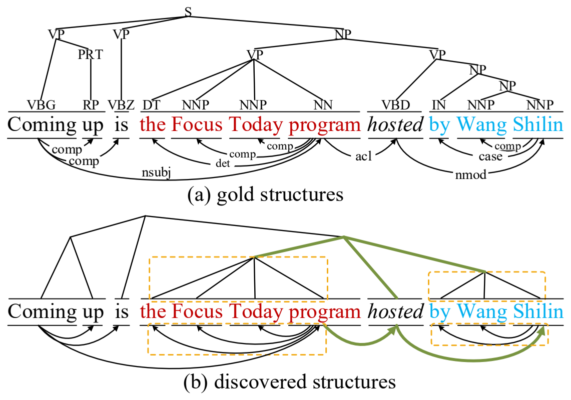
S
VP
VP
PRT
NP
VP
VP
NP
NP
VBG RP VBZ DT NNP NNP NN VBD IN NNP NNP
Coming up is the Focus Today program hosted by Wang Shilin
comp
comp
nsubj
det
comp
comp
acl
case
comp
nmod
(a) gold structures
Coming up is the Focus Today program hosted by Wang Shilin
(b) discovered structures

Figure 6: A SRL case where *hosted* is predicate, the Focus Today program is *A0*, by Wang Shilin is *A1*. Bold **green** lines indicates the edges with higher scores.

5 Conclusion

We investigated knowledge distillation on heterogeneous tree structures integration for facilitating NLP tasks, distilling syntactic knowledge into a sequential input encoder, in both output and feature level distillations. Results on four representative syntax-dependent tasks showed that the distilled student outperformed all standalone tree models, as well as the commonly used ensemble methods, indicating the effectiveness of the proposed method. Further analysis demonstrated that our method enjoys high robustness and efficiency.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 61772378, No. 61702121), the National Key Research and Development Program of China (No. 2017YFC1200500), the Research Foundation of Ministry of Education of China (No. 18JZD015), the Major Projects of the National Social Science Foundation of China (No. 11&ZD189), the Key Project of State Language Commission of China (No. ZDI135-112) and Guangdong Basic and Applied Basic Research Foundation of China (No. 2020A151501705).

References

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of EMNLP*, pages 632–642.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL*.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of ACL*, pages 173–180.

Junkun Chen, Xipeng Qiu, Pengfei Liu, and Xuanjing Huang. 2018. Meta multi-task learning for sequence modeling. In *Proceedings of AAAI*, pages 5070–5077.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *Proceedings of ACL*, pages 1657–1668.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Shiyu Wu, and Xuanjing Huang. 2015. Sentence modeling with gated recursive neural network. In *Proceedings of EMNLP*, pages 793–798.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of the Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111.

Kevin Clark, Minh-Thang Luong, Urvashi Khandelwal, Christopher D. Manning, and Quoc V. Le. 2019. BAM! born-again multi-task networks for natural language understanding. In *Proceedings of ACL*, pages 5931–5937.

John Cocke. 1970. Programming languages and their compilers: preliminary notes.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of ACL*, pages 16–23.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of ACL*, pages 2126–2136.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL*, pages 4171–4186.

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

Richárd Farkas, Bernd Bohnet, and Helmut Schmid. 2011. Features for phrase-structure reranking from dependency parses. In *Proceedings of ICPT*, pages 209–214.

Hao Fei, Yafeng Ren, and Donghong Ji. 2020a. Dispatched attention with multi-task learning for nested mention recognition. *Information Science*, 513:241–251.

Hao Fei, Yafeng Ren, and Donghong Ji. 2020b. A tree-based neural network model for biomedical event trigger detection. *Information Science*, 512:175–185.

Hao Fei, Meishan Zhang, Fei Li, and Donghong Ji. 2020c. Cross-lingual semantic role labeling with model transfer. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2427–2437.

Tommaso Furlanello, Zachary Chase Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. 2018. Born-again neural networks. In *Proceedings of ICML*, pages 1602–1611.

Ekaterina Garmash and Christof Monz. 2015. Bilingual structured language models for statistical machine translation. In *Proceedings of EMNLP*, pages 2398–2408.

Jetic Gū, Hassan S. Shavarani, and Anoop Sarkar. 2018. Top-down tree structured decoding with syntactic connections for neural machine translation and parsing. In *Proceedings of EMNLP*, pages 401–413.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. SemEval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of SemEval*, pages 33–38.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Ying Ju, Fubang Zhao, Shijie Chen, Bowen Zheng, Xuefeng Yang, and Yunfeng Liu. 2019. Technical report on conversational question answering. *CoRR*, abs/1909.10772.

Tadao Kasami. 1965. An efficient recognition and syntaxanalysis algorithm for context-free languages. *Technical Report Air Force Cambridge Research Lab*.

Yoshihide Kato and Shigeki Matsubara. 2019. PTB graph parsing with tree approximation. In *Proceedings of ACL*, pages 5344–5349.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of EMNLP*, pages 1317–1327.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of ICLR*.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of ACL*, pages 2676–2686.

Adhiguna Kuncoro, Chris Dyer, Laura Rimell, Stephen Clark, and Phil Blunsom. 2019. Scalable syntax-aware language models using knowledge distillation. In *Proceedings of ACL*, pages 3472–3484.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. In *Proceedings of IJCAI*, pages 2873–2879.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification. In *Proceedings of ACL*, pages 1–10.

Yang Liu, Matt Gardner, and Mirella Lapata. 2018. Structured alignment networks for matching sentences. In *Proceedings of EMNLP*, pages 1554–1564.

Moshe Looks, Marcello Herreshoff, DeLesley Hutchins, and Peter Norvig. 2017. Deep learning with dynamic computation graphs. In *Proceedings of ICLR*.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of EMNLP*, pages 1506–1515.

Thien Hai Nguyen and Kiyoaki Shirai. 2015. Phrasernn: Phrase recursive neural network for aspect-based sentiment analysis. In *Proceedings of EMNLP*, pages 2509–2514.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL*, pages 2227–2237.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using OntoNotes. In *Proceedings of CoNLL*, pages 143–152.

Xiaona Ren, Xiao Chen, and Chunyu Kit. 2013. Combine constituent and dependency parsing via reranking. In *Proceedings of IJCAI*, pages 2155–2161.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron C. Courville. 2018. Ordered neurons: Integrating tree structures into recurrent neural networks. *CoRR*, abs/1810.09536.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*, pages 1631–1642.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-informed self-attention for semantic role labeling. In *Proceedings of EMNLP*, pages 5027–5038.

Michalina Strzyz, David Vilares, and Carlos Gómez-Rodríguez. 2019. Sequence labeling parsing by learning across representations. In *Proceedings of ACL*, pages 5350–5357.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019. Patient knowledge distillation for BERT model compression. In *Proceedings of EMNLP*, pages 4322–4331.

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

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015b. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Zhiyang Teng and Yue Zhang. 2017. Head-lexicalized bidirectional tree lstms. *Transactions of the Association for Computational Linguistics*, 5:163–177.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NIPS*, pages 5998–6008.

Yogarshi Vyas and Marine Carpuat. 2019. Weakly supervised cross-lingual semantic relation classification via knowledge distillation. In *Proceedings of EMNLP*, pages 5284–5295.

David H. Wolpert. 1992. Stacked generalization. *Neural Networks*, 5(2):241–259.

Mingzhou Xu, Derek F. Wong, Baosong Yang, Yue Zhang, and Lidia S. Chao. 2019. Leveraging local and global patterns for self-attention networks. In *Proceedings of ACL*, pages 3069–3075.

Majid Yazdani and James Henderson. 2015. Incremental recurrent neural network dependency parser with search-based discriminative training. In *Proceedings of CoNLL*, pages 142–152.

Masashi Yoshikawa, Hiroshi Noji, and Yuji Matsumoto. 2017. A* CCG parsing with a supertag and dependency factored model. In *Proceedings of ACL*, pages 277–287.

Daniel H Younger. 1975. Recognition and parsing of context-free languages in time n3. *Information and Control*, 10(2):189–208.

Sergey Zagoruyko and Nikos Komodakis. 2017. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *Proceedings of ICLR*.

Xingxing Zhang, Liang Lu, and Mirella Lapata. 2016. Top-down tree long short-term memory networks. In *Proceedings of NAACL*, pages 310–320.

Yuan Zhang and Yue Zhang. 2019. Tree communication models for sentiment analysis. In *Proceedings of ACL*, pages 3518–3527.

Yue Zhang, Qi Liu, and Linfeng Song. 2018a. Sentence-state LSTM for text representation. In *Proceedings of ACL*, pages 317–327.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018b. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of EMNLP*, pages 2205–2215.

Ganbin Zhou, Ping Luo, Rongyu Cao, Yijun Xiao, Fen Lin, Bo Chen, and Qing He. 2017. Tree-structured neural machine for linguistics-aware sentence generation. *CoRR*, abs/1705.00321.

Junru Zhou and Hai Zhao. 2019. Head-driven phrase structure grammar parsing on Penn treebank. In *Proceedings of ACL*, pages 2396–2408.