Modularized Syntactic Neural Networks for Sentence Classification

Haiyan Wu, Ying Liu[*], Shaoyun Shi
Tsinghua University, Beijing, China, 100084
wuhy17@mails.tsinghua.edu.cn

Abstract

This paper focuses on tree-based modeling for the sentence classification task. In existing works, aggregating on a syntax tree usually considers local information of sub-trees. In contrast, in addition to the local information, our proposed Modularized Syntactic Neural Network (MSNN) utilizes the syntax category labels and takes advantage of the global context while modeling sub-trees. In MSNN, each node of a syntax tree is modeled by a label-related syntax module. Each syntax module aggregates the outputs of lower-level modules, and finally, the root module provides the sentence representation. We design a tree-parallel mini-batch strategy for efficient training and predicting. Experimental results on four benchmark datasets show that our MSNN significantly outperforms previous state-of-the-art tree-based methods on the sentence classification task.

1 Introduction

Text classification is an important and fundamental problem in natural language processing (NLP). With the increasing spread of the Internet, there are numerous applications of classification of short texts with only one sentence, for example, classifying questions according to what product or which part of the product architecture the question regards, sentiment analysis of customer reviews or tweets, and fast category detection based on news titles. Different from document classification, in which there are more topic words and features of writing styles, a single sentence contains limited information. Thus, understanding the meaning of a sentence is vital.

Although sequential models like long short term memory (LSTM) (Hochreiter and Schmidhuber, 1997) and gated recurrent units (GRU) (Cho et al., 2014b) have been widely used and provide excellent performances, it is hard for them to capture the syntactic information, which is essential for understanding sentences (Linzen et al., 2016). To utilize the syntactic information, some works proposed models taking parse trees or dependency trees as inputs (Le and Zuidema, 2015; Teng and Zhang, 2017; Socher et al., 2013; Zhu et al., 2015; Bowman et al., 2016). Previous researchers have empirically verified that these methods help to model sentences (Li et al., 2015). However, to improve the model efficiency and simplify the implementation, these methods binarize trees (Wang et al., 2007; Huang, 2007) so that they can be traversed by recursive neural networks (RvNN) or as a sequence by RNN. Although some models, such as Tree-LSTM (Tai et al., 2015; Looks et al., 2017; Ran and Zhong, 2019), theoretically support original parse trees, child nodes are simply summed, but the relationships among them are not modeled. The authors only conduct experiments on binary trees. Binary trees weaken the syntactic information and conceal the relationships among nodes at the same level or different levels. Latent tree is another way of modeling sentences, but it does not take full advantage of the syntactic information (Cho et al., 2014a; Choi et al., 2018; Williams et al., 2018; Addi et al., 2020). Current graph-based models specifically focus on the dependency tree (Marcheggiani and Titov, 2017; Zhang et al., 2018). Besides, these models do not consider the context information in the bottom-up aggregation. Syntax category labels are also not fully utilized. However, they both can affect the meanings of words and phrases, which should be considered.

In this work, a novel Modularized Syntactic Neural Network (MSNN) is proposed to model syntax trees of sentences. Each node in the tree is transformed into a syntax module in MSNN. The num-

[*] Corresponding Author: yingliu@mail.tsinghua.edu.cn

ber of distinct syntax modules is the same as the number of distinct syntax category labels. A category label is the syntactic category of a subtree's root, e.g. "NP", "VP", etc. The modules are used to build networks according to tree structures, and there is a one-to-one correspondence between tree structures and network structures. The parameters of modules corresponding to the same category labels are shared. Note that there is no limitation to binary trees, and our implementation of tree-parallel mini-batches based on the original parse trees provides excellent efficiency. Each syntax module aggregates outputs of lower-level modules and outputs a representation vector of the sub-tree. Syntax category labels and global context information are encoded to guide the propagation and better infer the meaning of the sentence. The root module finally outputs the sentence representation, which is further used for classification. We test MSNN on four benchmark datasets, and the results show that it outperforms previous state-of-the-art methods.

The main contributions of this work are listed as follows:

- A novel Modularized Syntactic Neural Network (MSNN) is proposed to model syntax trees for sentence classification. Both category labels and global context information are utilized when modeling sub-trees.
- We provide a design of tree-parallel mini-batches so that binarization of trees is not necessary and structural information is better reserved.
- Experimental results on four benchmark datasets show that MSNN significantly outperforms previous state-of-the-art methods on the sentence classification task.

2 Modularized Syntactic Neural Networks

An example of Modularized Syntactic Neural Networks (MSNN) is shown in Figure 1.

2.1 Global Context Bi-LSTM

Single words contain no sequential information. Meanings of words can be inferred from their context. It is essential to represent words in a certain context. A global context bidirectional LSTM (Bi-LSTM) (Schuster and Paliwal, 1997) is used to generate context-enhanced word vectors and global context vector. Suppose the input sentence s consists of a sequence of words $s = \{w_1, ..., w_t, ..., w_{|s|}\}$, where w_t is the t-th word in the sentence and $|s|$ is the sentence length. We use bold fonts to represent the vectors of words and other objects. The word vectors $\mathbf{w}_t \in \mathbb{R}^d$ can either be randomly initialized or pre-trained vectors, and d is the dimension of word vectors. To enrich word vectors with the context information in the sentence, a Bi-LSTM is applied to the sequence of words $\{w_t\}_{t=1...|s|}$. Let $\mathbf{h}_t^f$ denote the hidden states of the forward LSTM at position t, in which the past context information is included. By another backward LSTM, hidden states containing the future context $\mathbf{h}_t^b$ are formed. The initial hidden states $\mathbf{h}_0$ are zero-initialized. Then the enriched word vector $\mathbf{e}_t$ at position t is

$$\mathbf{e}_t = \mathbf{h}_t^f + \mathbf{h}_t^b + \mathbf{w}_t \tag{1}$$

The global context vector $\mathbf{c}_s$ of sentence s is

$$\mathbf{c}_s = \mathbf{h}_{|s|}^f + \mathbf{h}_1^b \tag{2}$$

$\mathbf{e}_t$ and $\mathbf{c}_s$ are inputs of syntax modules. Global context information is embedded into $\mathbf{e}_t$, so that the information propagation in higher layers is guided by the context.

2.2 Syntax Modules

Previous works use a constant structure to traverse over trees, which do not consider the diversity of syntax category labels. In MSNN, syntax modules are used to construct different network structures according to the syntax trees. Each category label (including POS tags of leaf nodes) l in the syntax tree is mapped to a module $M_l(\cdot)$. Each module takes output vectors of its child nodes as inputs, and it outputs the representation of the sub-tree. The number of distinct syntax modules is the same as the number of distinct syntax category labels. For example, in Figure 1, seven different modules are used to assemble the network: "S, NP, VP, ., PRP$, NN, VBZ". According to whether a node is a leaf, modules are divided into two categories: leaf POS module and root/intermediate module.

Each word in the sentence w_t has a POS p_t, which shows the categories of words according to their function in a sentence. Words with the same POS have similar syntactic behaviors. In POS module, the enhanced word vector $\mathbf{e}_t$ and the

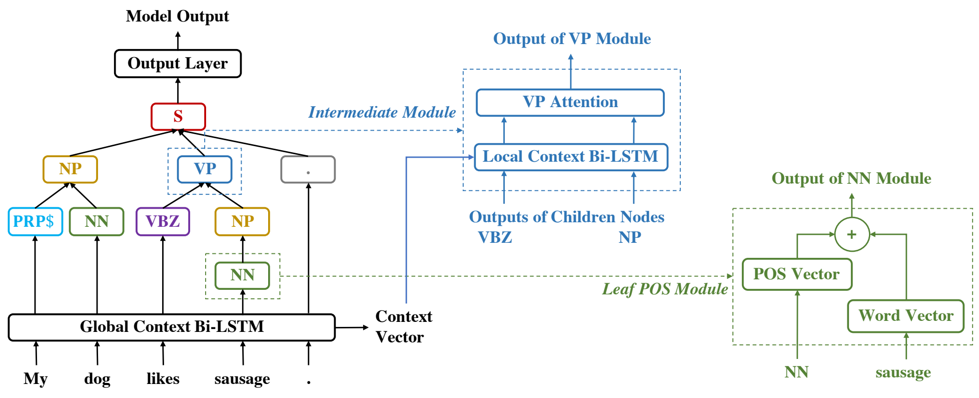

Figure 1: An example of MSNN.

POS vector $\mathbf{p}_t \in \mathbb{R}^d$ are combined:

$$M_{p_t}(\mathbf{e}_t, \mathbf{p}_t) = \mathbf{e}_t + \mathbf{p}_t = \mathbf{m}_{p_t} \tag{3}$$

Vectors of POS labels are randomly initialized and learned during the training. The output $\mathbf{m}_{p_t}$ will be the input of its parent module.

It is necessary to model relationships among sibling nodes because the information from syntactic nodes' sisterhood may reveal useful for sentence classification. Examples may be negation clauses or modifiers. When modeling the sentence as a sequence, it is not easy for RNN, CNN, or other structures to identify their lexical scope. And in a binary tree, coordinate relations among nodes are diluted. The influence of negation clauses or modifiers on some nodes may be hard to capture, especially in long phrases and sentences. In MSNN, suppose a root or intermediate module $M_l(\cdot)$ with label l have n child modules $\{M_{c_1}, M_{c_2}, ..., M_{c_n}\}$, and their output vectors are $\{\mathbf{m}_{c_1}, \mathbf{m}_{c_2}, ..., \mathbf{m}_{c_n}\}$. Then in the $M_l(\cdot)$, firstly, these vectors are modeled by a local context Bi-LSTM:

$$\textit{Bi-LSTM}(\mathbf{c}_s, \mathbf{m}_{c_1}...\mathbf{m}_{c_n}) = \{\mathbf{e}_{c_1}, \mathbf{e}_{c_2}, ..., \mathbf{e}_{c_n}\} \tag{4}$$

where *Bi-LSTM*$(\cdot)$ have the similar structure as the global context Bi-LSTM in section 2.1 but different parameters. The local context Bi-LSTM shares the same parameters among different modules, and the outputs $\mathbf{e}_{c_1}...\mathbf{e}_{c_n}$ are the enriched representations of child nodes. The global context vector $\mathbf{c}_s$ is used to initialize the hidden state and cell state in order to guide the information to propagate in the local syntactic node. The context can affect the semantic meanings of phrases, e.g., representations of syntactic nodes.

Different child nodes contribute more or less to the representation of their parent node. A syntax-aware attention network is then used to aggregate child nodes:

$$\mathbf{k}_{c_i} = \delta(\mathbf{K}\mathbf{e}_{c_i} + \mathbf{b}_k),\ \mathbf{q}_l = \delta(\mathbf{Q}_l(\mathbf{l} \oplus \mathbf{c}_s) + \mathbf{b}_l),$$
$$\mathbf{m}_l = \sum_{j=1}^{n} a_{c_j}\mathbf{e}_{c_j},\ a_{c_i} = \frac{exp(\mathbf{q}_l^\top \mathbf{k}_{c_i})}{\sum_{j=1}^{n} exp(\mathbf{q}_l^\top \mathbf{k}_{c_j})} \tag{5}$$

where $\mathbf{K} \in \mathbb{R}^{d\times d}$ is the global transformation weight matrix for attention keys, and $\mathbf{b}_k \in \mathbb{R}^d$ is the bias vector. Label-related query vector $\mathbf{q}_l$ is used to evaluate whether children are informative for the parent node in such sentence context $\mathbf{c}_s$ and category label $\mathbf{l}$. $\mathbf{Q}_l \in \mathbb{R}^{d\times d}$ is the query transformation weight matrix of category label l, $\mathbf{b}_l$ is the bias vector, and $\mathbf{l} \in \mathbb{R}^d$ is the vector of category label. They are all label-related parameters so that syntax modules of labels have different parameters. $\delta(\cdot)$ is the non-linear activation function and we use the LeakyRelu (Maas et al., 2013). a_{c_i} is the normalized attention weights by a softmax layer. The l-module outputs the representation of the sub-tree $\mathbf{m}_l$ by a weighted sum. In this way, context and syntactic information guide the information to propagate in sub-trees.

The aggregation process goes from bottom to up, and finally, the root module outputs the sentence vector $\mathbf{m}_S$ for further classification.

A fully-connected layer followed by a softmax function is used to give the final predictions of classification. The Cross-Entropy with ℓ_2-regularization is the loss function to train the model.

2.3 Tree-Parallel Mini-Batch

Tree structures of sentences vary a lot. As a deep model, it is essential to construct mini-batches for effective and efficient training and testing. Previous tree-based models usually construct binary

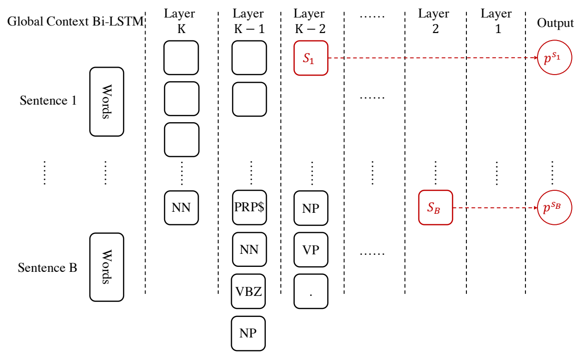

Figure 2: Brief illustration of a tree-parallel mini-batch.

trees to simplify the implementation. However, to form the binarized tree, many intermediate nodes are inserted in the original tree. Some nodes or phrases with coordinate relation in the raw sentences or the original parse trees may now on different levels of binary trees, and their paths and path-lengths to the root vary a lot. In such a situation, the parent-children or brother-sister relationships among nodes are captured implicitly in the binary tree. It is not easy to design good ways to construct features encoding information about node sisterhood. In contrast, we design a way of tree-parallel mini-batches for MSNN, as shown in Figure 2. B is the batch size, e.g., the number of sentences in the batch. K is the maximum number of layers of all trees in the batch. In a batch running, the first step is all sentences going through the global context Bi-LSTM in parallel to obtain enriched word vectors. In the second step, all nodes on the last layer of different trees are calculated simultaneously. And then the last but one layer, and so on. As long as previous layers have been calculated, the required information of current layers is all available. For example, the tree in Figure 1 is shown as the B-th sentence in Figure 2. The number of iterations along layers depends on the maximum depth of trees in the batch. Finally, outputs of all root nodes are gathered for the output layer.

3 Experiments

3.1 Datasets and Settings

We use four text classification datasets from Zhang et al. (2015), including AG's News, DBpedia, Amazon Review Polarity (ARP), Amazon Review Full (ARF). For the sentence classification task, only single-sentence instances are used in each dataset. Some detailed information of datasets is shown in Table 1.

Dataset	#Train	#Valid	#Test	#Class
AG's News	60K	10K	10K	4
DBpedia	70K	7K	7K	14
ARP	400K	80K	80K	2
ARF	500K	100K	100K	5

Table 1: Statistics of evaluation datasets.

MSNN is compared with two sequential models **LSTM** (Hochreiter and Schmidhuber, 1997), **Bi-LSTM** (Schuster and Paliwal, 1997) and two tree-based methods **Tree-LSTM** (Tai et al., 2015), **Gumbel-Tree** (Choi et al., 2018). We use the PyTorch implementation provided by Shi et al. (2018).

All models, including baselines, are trained with Adam (Kingma and Ba, 2014) in mini-batches at the size of 64. The learning rate is 1×10^{-4}, and early-stopping is conducted according to the performance on the validation set. The weight of ℓ_2-regularization λ is manually searched between $\{0, 1\times10^{-5}, 1\times10^{-4}, 1\times10^{-3}, 1\times10^{-2}\}$. Word vectors are randomly initialized. The dimension of word vectors and hidden layers d is 300. We run the experiments with 5 different random seeds and report the average accuracy and standard errors. All models are trained with a GPU (NVIDIA GeForce GTX 1080Ti).

3.2 Results and Discussion

The overall performances are shown in Table 2, in which "w/o" means "without". We can conclude that tree-based models like Tree-LSTM and Gumbel-Tree are better than sequential models LSTM and Bi-LSTM, which shows the superiority of modeling structures of the syntax tree for sentence classification. Gumbel-Tree is slightly better than Tree-LSTM because of a more flexible structure and a similar global context RNN as MSNN. However, it uses the Gumbel softmax (Jang et al., 2017) to form latent trees and does not take full advantage of the syntactic structure information. Although Gumbel-Tree is more flexible in integrating context information and constructing sentence representations, it produces unstable latent trees (Williams et al., 2018). MSNN outperforms these baselines because it utilizes the global context

Detailed implementation can be found at `https://github.com/wuhaiyan2014/MSNN`.

Model	AG's News	DBpedia	ARP	ARF
LSTM	0.9050 ± 0.0021	0.9550 ± 0.0011	0.8887 ± 0.0003	0.5083 ± 0.0006
Bi-LSTM	0.9033 ± 0.0014	0.9517 ± 0.0015	0.8914 ± 0.0004	0.5111 ± 0.0002
Tree-LSTM	0.9107 ± 0.0008	0.9639 ± 0.0027	0.8913 ± 0.0002	0.5142 ± 0.0003
Gumbel-Tree	0.9111 ± 0.0006	0.9582 ± 0.0011	0.8914 ± 0.0004	0.5158 ± 0.0004
MSNN	**0.9173** ± 0.0008	**0.9797** ± 0.0003	**0.8916** ± 0.0006	**0.5185** ± 0.0011
w/o global RNN	0.9164 ± 0.0010	0.9784 ± 0.0006	0.8875 ± 0.0004	0.5137 ± 0.0002
w/o local RNN	0.9170 ± 0.0006	0.9769 ± 0.0009	0.8885 ± 0.0003	0.5134 ± 0.0005
w/o attention	0.9072 ± 0.0013	0.9741 ± 0.0004	0.8906 ± 0.0005	0.5172 ± 0.0006
w/o category label	0.9145 ± 0.0008	0.9774 ± 0.0003	0.8894 ± 0.0002	0.5134 ± 0.0006

Table 2: Overall performance on four benchmark datasets.

and syntax category labels to guide the information propagation in sub-trees. The meaning of words and phrases can be inferred by their context and syntactic roles. Besides, MSNN based on the tree-parallel mini-batch design is not limited to binary trees, which fully retains the syntactic information. The local RNN and attention network capture the relationship between nodes with the same parent. The improvements of MSNN are larger on DBpedia than that on other datasets. The reason is that most of the sentences in the DBpedia are high-quality declarative sentences, and their tree structures are less complex compared to reviews in Amazon datasets, which contain much noise. Clean syntactic information on DBpedia results in wider discrepancy, not only between MSNN and Gumbel-Tree but also between tree-based methods and Bi-LSTM.

To study the ablation of different parts, we remove the global RNN, local RNN, attention mechanism, or category label information in MSNN. Results show that all these parts contribute to the excellent performance of MSNN. Global RNN enriches word vectors with context information and provides a global context representation of the sentence. Local RNN captures the relationships among nodes on the same level. The attention mechanism dynamically aggregates information from child nodes under the guidance of context and syntax category labels. Category labels also show the roles of words and phrases in the sentence.

The average training time per epoch on the largest ARF dataset under the same validating frequency is shown in Table 3. Generally, sequential methods are much faster than tree-based models because of simple computation graphs. With the help of tree-parallel mini-batch, MSNN largely reduces

Model	Time/epoch
LSTM	888s
Bi-LSTM	972s
Tree-LSTM	4,212s
Gumbel-Tree	4,908s
MSNN	3,707s

Table 3: Running time per training epoch on ARF.

redundant calculations and is more efficient compared with Gumbel-Tree and Tree-LSTM. Binary trees usually have many more nodes than original trees. Traverse them node by node is slower than calculating nodes in parallel. Besides, directly modeling of origin trees largely retains the structural information.

4 Conclusion

In this work, a novel model MSNN is proposed to model syntax trees for classification. It uses global context information and syntax category labels to help improve the modeling of sub-trees and thus better sentence representations. A tree-parallel mini-batch strategy is further designed for efficient running and support for non-binary trees. Our future work will include conducting experiments on dependency trees and more NLP tasks.

Acknowledgments

This work is supported by Project of Humanities and Social Sciences of Ministry of Education in China (17YJAZH056), 2018 National Major Program of Philosophy and Social Science Fund (18ZDA238), Tsinghua University Humanities and Social Sciences Revitalization Project (2019THZWJC38).

References

Hajar Ait Addi, Redouane Ezzahir, and Abdelhak Mahmoudi. 2020. Three-level binary tree structure for sentiment classification in arabic text. In *Proceedings of the 3rd International Conference on Networking, Information Systems & Security*, pages 1–8.

Samuel Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1466–1477.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734.

Jihun Choi, Kang Min Yoo, and Sang-goo Lee. 2018. Learning to compose task-specific tree structures. In *32nd AAAI Conference on Artificial Intelligence*.

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

Liang Huang. 2007. Binarization, synchronous binarization, and target-side binarization. In *Proceedings of SSST, NAACL-HLT 2007/AMTA Workshop on Syntax and Structure in Statistical Translation*, pages 33–40.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparametrization with gumble-softmax. In *International Conference on Learning Representations (ICLR 2017)*. OpenReview. net.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Phong Le and Willem Zuidema. 2015. Compositional distributional semantics with long short term memory. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 10–19.

Jiwei Li, Minh-Thang Luong, Dan Jurafsky, and Eduard Hovy. 2015. When are tree structures necessary for deep learning of representations? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2304–2314.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Moshe Looks, Marcello Herreshoff, DeLesley Hutchins, and Peter Norvig. 2017. Deep learning with dynamic computation graphs. *International Conference on Learning Representations (ICLR 2017)*.

Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1506–1515.

Wang Ran and Jin Zhong. 2019. Tree-structured networks based on polarity shifting and lstm for sentences classification. *Application Research of Computers*, 1:15.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Haoyue Shi, Hao Zhou, Jiaze Chen, and Lei Li. 2018. On tree-based neural sentence modeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4631–4641.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1556–1566.

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

Wei Wang, Kevin Knight, and Daniel Marcu. 2007. Binarizing syntax trees to improve syntax-based machine translation accuracy. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 746–754.

Adina Williams, Andrew Drozdov*, and Samuel R Bowman. 2018. Do latent tree learning models identify meaningful structure in sentences? *Transactions of the Association for Computational Linguistics*, 6:253–267.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Yuhao Zhang, Peng Qi, and Christopher D Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2205–2215.

Xiaodan Zhu, Parinaz Sobihani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *International Conference on Machine Learning*, pages 1604–1612.