You Only Need Attention to Traverse Trees

Mahtab Ahmed, Muhammad Rifayat Samee and Robert E. Mercer

Department of Computer Science, University of Western Ontario
{mahme255, msamee, rmercer}@uwo.ca

Abstract

In recent NLP research, a topic of interest is universal sentence encoding, sentence representations that can be used in any supervised task. At the word sequence level, fully attention-based models suffer from two problems: a quadratic increase in memory consumption with respect to the sentence length and an inability to capture and use syntactic information. Recursive neural nets can extract very good syntactic information by traversing a tree structure. To this end, we propose *Tree Transformer*, a model that captures phrase level syntax for constituency trees as well as word-level dependencies for dependency trees by doing recursive traversal only with attention. Evaluation of this model on four tasks gets noteworthy results compared to the standard transformer and LSTM-based models as well as tree-structured LSTMs. Ablation studies to find whether positional information is inherently encoded in the trees and which type of attention is suitable for doing the recursive traversal are provided.

1 Introduction

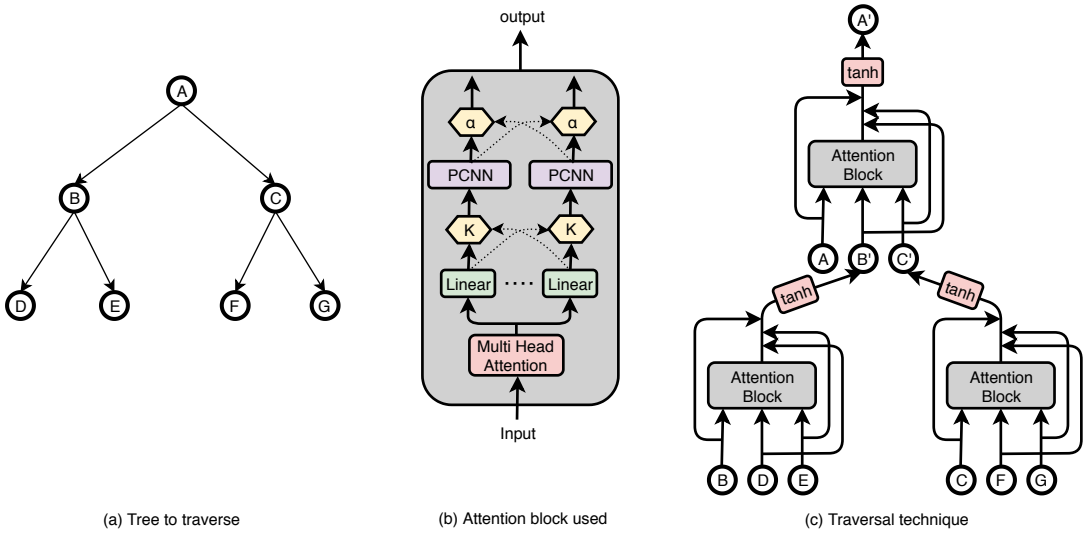
Following the breakthrough in NLP research with word embeddings by Mikolov et al. (2013), recent research has focused on sentence representations. Having good sentence representations can help accomplish many NLP tasks because we eventually deal with sentences, e.g., question answering, sentiment analysis, semantic similarity, and natural language inference.

Most of the existing task specific sequential sentence encoders are based on recurrent neural nets such as LSTMs or GRUs (Conneau et al., 2017; Lin et al., 2017; Liu et al., 2016). All of these works follow a common paradigm: use an LSTM/GRU over the word sequence, extract contextual features at each time step, and apply some kind of pooling on top of that. However, a few works adopt some different methods. Kiros et al. (2015) propose a skip-gram-like objective function at the sentence level to obtain the sentence embeddings. Logeswaran and Lee (2018) reformulate the task of predicting the next sentence given the current one into a classification problem where instead of a decoder they use a classifier to predict the next sentence from a set of candidates.

The attention mechanism adopted by most of the RNN based models require access to the hidden states at every time step (Yang et al., 2016; Kumar et al., 2016). These models are inefficient and at the same time very hard to parallelize. To overcome this, Parikh et al. (2016) propose a fully attention-based neural network which can adequately model the word dependencies and at the same time is parallelizable. Vaswani et al. (2017) adopt the multi-head version in both the encoder and decoder of their Transformer model along with positional encoding. Ahmed et al. (2017) propose a multi-branch attention framework where each branch captures a different semantic subspace and the model learns to combine them during training. Cer et al. (2018) propose an unsupervised sentence encoder by leveraging only the encoder part of the Transformer where they train on the large Stanford Natural Language Inference (SNLI) corpus and then use transfer learning on smaller task specific corpora.

Apart from these sequential models, there has been extensive work done on the tree structure of natural language sentences. Socher et al. (2011b, 2013, 2014) propose a family of recursive neural net (RvNN) based models where a composition function is applied recursively bottom-up on children nodes to compute the parent node representation until the root is reached. Tai et al. (2015) propose two variants of sequential LSTM, child sum tree LSTM and N-ary tree LSTM. The same gating structures as in standard LSTM are used except

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 316–322
Florence, Italy, July 28 - August 2, 2019.



(a) Tree to traverse (b) Attention block used (c) Traversal technique

Figure 1: Attention over the tree structure

the hidden and cell states of a parent are dependent only on the hidden and cell states of its children.

Recently, Shen et al. (2018) propose a Parsing-Reading-Predict Network (PRPN) which can induce syntactic structure automatically from an unannotated corpus and can learn a better language model with that induced structure. Later, Htut et al. (2018) test this PRPN under various configurations and datasets and further verified its empirical success for neural network latent tree learning. Williams et al. (2018) also validate the effectiveness of two latent tree based models but found some issues such as being biased towards producing shallow trees, inconsistencies during negation handling, and a tendency to consider the last two words of a sentence as constituents.

In this paper, we propose a novel recursive neural network architecture consisting of a decomposable attention framework in every branch. We call this model *Tree Transformer* as it is solely dependent on attention. In a subtree, the use of a composition function is justified by a claim of Socher et al. (2011b, 2014). In this work, we replace this composition function with an attention module. While Socher et al. (2011b, 2014) consider only the child representations for both dependency and constituency syntax trees, in this work, for dependency trees, the attention module takes both the child and parent representations as input and produces weighted attentive copies of them. For constituency trees, as the parent vector is entirely dependent on the upward propagation, the attention module works only with the child representations. Our extensive evaluation proves that our model is better or at least on par with the existing sequential (i.e., LSTM and Transformer) and tree structured (i.e., Tree LSTM and RvNN) models.

2 Proposed Model

Our model is designed to address the following general problem. Given a dependency or constituency tree structure, the task is to traverse every subtree within it attentively and infer the root representation as a vector. Our idea is inspired by the RvNN models from Socher et al. (2013, 2011b, 2014) where a composition function is used to transform a set of child representations into one single parent representation. In this section, we describe how we use the attention module as a composition function to build our *Tree Transformer*. Figure 1 gives a sketch of our model.

A dependency tree contains a word at every node. To traverse a subtree in a dependency tree, we look at both the parent and child representations ($\mathbf{X_d}$ in Eqn. 1). In contrast, in a constituency tree, only leaf nodes contain words. The non-terminal vectors are calculated only after traversing each subtree. Consequently, only the child representations ($\mathbf{X_c}$ in Eqn. 1) are considered.

$$\mathbf{X_d} = \begin{bmatrix} \mathbf{p_v} \\ \mathbf{c_{1_v}} \\ \vdots \\ \mathbf{c_{n_v}} \end{bmatrix} \qquad \mathbf{X_c} = \begin{bmatrix} \mathbf{c_{1_v}} \\ \mathbf{c_{2_v}} \\ \vdots \\ \mathbf{c_{n_v}} \end{bmatrix} \tag{1}$$

Here, $\mathbf{p_v}$ is the parent representation and the $\mathbf{c}_{i_v}$'s are the child representations. For both of these trees, Eqn. 2 computes the attentive transformed

representation.

$$\tilde{\mathbf{P}} = \mathbf{f}(\mathbf{x}), \quad \textit{where } \mathbf{x} \in \{\mathbf{X_d}, \mathbf{X_c}\} \tag{2}$$

Here, $\mathbf{f}$ is the composition function using the multi-branch attention framework (Ahmed et al., 2017). This multi-branch attention is built upon the multi-head attention framework (Vaswani et al., 2017) which further uses scaled dot-product attention (Parikh et al., 2016) as the building block. It operates on a query $\mathbf{Q}$, key $\mathbf{K}$ and value $\mathbf{V}$ as follows

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\mathbf{T}}{\sqrt{\mathbf{d}_k}}\right)\mathbf{V} \tag{3}$$

where $\mathbf{d}_k$ is the dimension of the key. As we are interested in n branches, n copies are created for each (Q, K, V), converted to a 3D tensor, and then a scaled dot-product attention is applied using

$$\mathbf{B}_i = \text{Attention}(\mathbf{Q}_i\mathbf{W}_i^Q, \mathbf{K}_i\mathbf{W}_i^K, \mathbf{V}_i\mathbf{W}_i^V) \tag{4}$$

where $i \in [1, n]$ and the $\mathbf{W}_i$'s are the parameters that are learned. Note that $\mathbf{W}_i^Q, \mathbf{W}_i^K$ and $\mathbf{W}_i^V \in \mathbb{R}^{d_m \times d_k}$. Instead of having separate parameters for the transformation of leaves, internal nodes and parents (Socher et al., 2014), we keep $\mathbf{W}_i^Q, \mathbf{W}_i^K$ and $\mathbf{W}_i^V$ the same for all these components. We then project each of the resultant tensors into different semantic sub-spaces and employ a residual connection (He et al., 2016; Srivastava et al., 2015) around them. Lastly, we normalize the resultant outputs using a layer normalization block (Ba et al., 2016) and apply a scaling factor κ to get the branch representation. All of these are summarized in Eqn. 5.

$$\overline{\mathbf{B}}_i = \text{LayerNorm}(\mathbf{B}_i\mathbf{W}_i^b + \mathbf{B}_i) \times \kappa_i \tag{5}$$

Here, $\mathbf{W}_i^b \in \mathbb{R}^{n \times d_v \times d_m}$ and $\kappa \in \mathbb{R}^n$ are the parameters to be learned. Note that we choose $\mathbf{d}_k = \mathbf{d}_q = \mathbf{d}_v = \mathbf{d}_m/n$. Following this, we take each of these $\overline{\mathbf{B}}$'s and apply a convolutional neural network (see Eqn. 6) consisting of two transformations on each position separately and identically with a *ReLU* activation (R) in between.

$$\text{PCNN}(x) = \text{Conv}(\text{R}(\text{Conv}(\mathbf{x}) + \boldsymbol{b}_1)) + \boldsymbol{b}_2 \tag{6}$$

We compute the final attentive representation of these subspace semantics by doing a linearly weighted summation (see Eqn. 7) where $\alpha \in \mathbb{R}^n$ is learned as a model parameter.

$$\text{BranchAttn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \sum_{i=1}^{n} \alpha_i \text{PCNN}(\overline{\mathbf{B}}_i) \tag{7}$$

Lastly, we employ another residual connection with the output of Eqn. 7, transform it non-linearly and perform an element-wise summation (EwS) to get the final parent representation as in Eqn. 8.

$$\tilde{\mathbf{P}} = \text{EwS}(\tanh((\tilde{\mathbf{x}} + \mathbf{x})\mathbf{W} + b)) \tag{8}$$

Here, $\mathbf{x}$ and $\tilde{\mathbf{x}}$ depict the input and output of the attention module.

3 Experiments

In this section, we present the effectiveness of our *Tree Transformer* model by reporting its evaluation on four NLP tasks. We present a detailed ablation study on whether positional encoding is important for trees and also demonstrate which attention module is most suitable as a composition function for the recursive architectures.

Experimental Setup: We initialize the word embedding layer weights with GloVe 300-dimensional word vectors (Pennington et al., 2014). These embedding weights are not updated during training. In the multi-head attention block, the dimension of the query, key and value matrices are set to 50 and we use 6 parallel heads on each input. The multi-branch attention block is composed of 6 position-wise convolutional layers. The number of branches is also set to 6. We use two layers of convolutional neural network as the composition function for the PCNN layer. The first layer uses 341 $1d$ kernels with no dropout and the second layer uses 300 $1d$ kernels with dropout 0.1.

During training, the model parameters are updated using the Adagrad algorithm (Duchi et al., 2011) with a fixed learning rate of 0.0002. We trained our model on an Nvidia GeForce GTX 1080 GPU and used PyTorch 0.4 for the implementation under the Linux environment.

Datasets: Evaluation is done on four tasks: the Stanford Sentiment Treebank (SST) (Socher et al., 2011b) for sentiment analysis, Sentences Involving Compositional Knowledge (SICK) (Marelli et al., 2014) for semantic relatedness (-R) and natural language inference (-E), and the Microsoft Research Paraphrase (MSRP) corpus (Dolan et al., 2004) for paraphrase identification.

The samples in the SST dataset are labelled for both the binary and the 5-class classification task. In this work we are using only the binary classification labels. The MSRP dataset is labelled with two classes. The samples in the SICK dataset are labelled for both the 3-class SICK-E classification

Types of Models	Model	SICK-E (Acc.)	SICK-R (MSE)	SST (Acc.)	MSRP (Acc.)
Tree Structured	SDT-RNN (Socher et al., 2014)	-	.3848	-	-
	RAE (Socher et al., 2011a)	-	-	82.40	76.80
	MV-RNN (Socher et al., 2012)	58.14 †	-	82.90	66.91 †
	RNTN (Socher et al., 2013)	59.42 †	-	85.40	66.91 †
	DT-RNN (Socher et al., 2014)	63.38 †	.3822	86.60	67.51 †
	DT-LSTM (Tai et al., 2015)	83.11 †	.2532/.2625 †	85.70/85.10 †	72.07 †
	CT-LSTM (Tai et al., 2015)	82.00 †	.2734/.2891 †	88.00/87.27 †	70.07 †
LSTM	LSTM (Tai et al., 2015)	76.80	.2831	84.90	71.70
	Bi-LSTM (Tai et al., 2015)	82.11 †	.2736	87.50	72.70
	2-layer LSTM (Tai et al., 2015)	78.54 †	.2838	86.30	69.35 †
	2-layer Bi-LSTM (Tai et al., 2015)	79.66 †	.2762	87.20	70.40 †
	Infersent (Conneau et al., 2017)	84.62	.2732	86.00	74.46
Transformer	USE_T (Cer et al., 2018)	81.15	.5241 †	85.38	74.96 †
	USE_T+DAN (Cer et al., 2018)	-	-	86.62	-
	USE_T+CNN (Cer et al., 2018)	-	-	86.69	-
Tree Transformer	Dependency Tree Transformer (DTT)	82.95	.2774	83.12	70.34
	Constituency Tree Transformer (CTT)	82.72	.3012	86.66	71.73

Table 1: Performance comparison of the *Tree Transformer* against some state-of-the-art sentence encoders. Models that we implemented are marked with †.

task and the SICK-R regression task which uses real-valued labels between 1 and 5. Instead of doing a regression on SICK-R to predict the score, we are using the same setup as Tai et al. (2015) who compute a target distribution p as a function of the predicted score y given by Eqn. 9.

$$\tilde{p_i} = \begin{cases} y - \lfloor y \rfloor, & \text{if } i = \lfloor y \rfloor + 1 \\ \lfloor y \rfloor - y + 1, & \text{if } i = \lfloor y \rfloor \\ 0, & \text{otherwise} \end{cases} \quad (9)$$

The SST dataset includes already generated dependency and constituency trees. As the other two datasets do not provide tree structures, we parsed each sentence using the Stanford dependency and constituency parser (Manning et al., 2014).

For the sentiment classification (SST), natural language inference (SICK-E), and paraphrase identification (MSRP) tasks, accuracy, the standard evaluation metric, is used. For the semantic relatedness task (SICK-R), we are using mean squared error (MSE) as the evaluation metric.

We use KL-divergence as the loss function for SICK-R to measure the distance between the predicted and target distribution. For the other three tasks, we use cross entropy as the loss function.

Table 1 shows the results of the evaluation of the model on the four tasks in terms of task specific evaluation metrics. We compare our *Tree Transformer* against tree structured RvNNs, LSTM based, and Transformer based architectures.

To do a fair comparison, we implemented both variants of Tree LSTM and Transformer based architectures and some of the RvNN and LSTM based models which do not have reported results for every task. Instead of assessing on transfer performance, the evaluation is performed on each corpus separately following the standard train/test/valid split.

For SICK-E, our model achieved 82.95% and 82.72% accuracy with dependency and constituency tree, respectively, which is on par with DT-LSTM (83.11%) as well as CT-LSTM (82.00%) and somewhat better than the standard Transformer (81.15%). As can be seen, all of the previous recursive architectures are somewhat inferior to the *Tree Transformer* results.

For SICK-R, we are getting .2774 and .3012 MSE whereas the reported MSE for DT-LSTM and CT-LSTM are .2532 and .2734, respectively. However, in our implementation of those models with the same hyperparameters, we haven't been able to reproduce the reported results. Instead we ended up getting .2625 and .2891 MSE for DT-LSTM and CT-LSTM, respectively. On this task, our model is doing significantly better than the standard Transformer (.5241 MSE).

On the SST dataset, our model (86.66% Acc.) is again on par with tree LSTM (87.27% Acc.) and better than Transformer (85.38% Acc.) as well as Infersent (86.00% Acc.)[1].

On the MSRP dataset, our dependency tree version (70.34% Acc.) is below DT-LSTM (72.07%

[1]The official implementation available at https://github.com/facebookresearch/InferSent is used. Reported hyperparameters are used except LSTM hidden state, 1024d is chosen due to hardware limitations.

Model	PE	SICK-E	SICK-R	SST	MSRP
DTT	On	78.58	.3383	83.03	69.01
	Off	82.28	.2774	83.12	70.34
CTT	On	81.83	.3088	83.96	71.73
	Off	82.72	.3012	86.66	68.62

Table 2: Effect of Positional Encoding (PE).

Acc.). However, for the constituency tree version, we are getting better accuracy (71.73%) than CT-LSTM (70.07%). It is to be noted that all of the sequential models, i.e., Transformer, Infersent and LSTMs, are doing better compared to the tree structured models on this paraphrase identification task.

Model	S/M/B	SICK-E	SICK-R	SST	MSRP
DTT	S	82.95	.3004	81.71	68.62
	M	82.86	.2955	82.97	69.07
	B	82.28	.2774	83.12	70.34
CTT	S	80.17	.4657	84.58	69.35
	M	79.66	.4346	83.74	70.01
	B	82.72	.3012	86.32	71.73

Table 3: Effect of different attention modules as a composition function. **S**: single-head attention, **M**: multi-head attention, **B**: multi-branch attention.

Since positional encoding is a crucial part of the standard Transformer, Table 2 presents its effect on trees. In constituency trees, positional information is inherently encoded in the tree structure. However, this is not the case with dependency trees. Nonetheless, our experiments suggest that for trees, positional encoding is irrelevant information as the performance drops in all but one case. We also did an experiment to see which attention module is best suited as a composition function and report the results in Table 3. As can be seen, in almost all the cases, multi-branch attention has much better performance compared to the other two. This gain by multi-branch attention is much more significant for CTT than for DTT.

Figure 2 visualizes how our CTT model puts attention on different phrases in a tree to compute the correct sentiment. Space limitations allow only portions of the tree to be visualized. As can be seen, the sentiment is positive (+1) at the root and the model puts more attention on the right branch as it has all of the positive words, whereas the left branch (NP) is neutral (0). The bottom three trees are the phrases which contain the positive words. The model again puts more attention on the relevant branches. The words 'well' and 'sincere' are inherently positive. In the corpus the word 'us' is tagged as positive for this sentence.

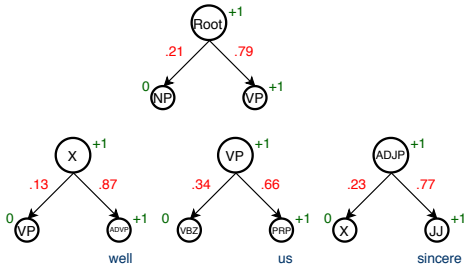


Figure 2: Attentive tree visualization (CTT)

4 Conclusion

In this paper, we propose *Tree Transformer* which successfully encodes natural language grammar trees utilizing the modules designed for the standard Transformer. We show that we can effectively use the attention module as the composition function together with grammar information instead of just bag of words and can achieve performance on par with Tree LSTMs and even better performance than the standard Transformer.

Acknowledgements

This research is partially funded by The Natural Sciences and Engineering Research Council of Canada (NSERC) through a Discovery Grant to Robert E. Mercer. We also acknowledge the helpful comments provided by the reviewers.

References

Karim Ahmed, Nitish Shirish Keskar, and Richard Socher. 2017. Weighted transformer network for machine translation. *arXiv preprint arXiv:1711.02132*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiacob, Rhomni St John, Noah Constant, Mario Guajardo-Céspedes, Steve Yuanc, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 350–356.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

Phu Mon Htut, Kyunghyun Cho, and Samuel Bowman. 2018. Grammar induction with neural language models: An unusual replication. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 371–373.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems*, pages 3294–3302.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning*, pages 1378–1387.

Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In *5th International Conference on Learning Representations (ICLR) Conference Track Proceedings*.

Yang Liu, Chengjie Sun, Lei Lin, and Xiaolong Wang. 2016. Learning natural language inference using bidirectional lstm model and inner-attention. *arXiv preprint arXiv:1605.09090*.

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. In *6th International Conference on Learning Representations (ICLR) Conference Track Proceedings*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 1–8.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Yikang Shen, Zhouhan Lin, Chin-Wei Huang, and Aaron Courville. 2018. Neural language modeling by jointly learning syntax and lexicon. In *6th International Conference on Learning Representations (ICLR) Conference Track Proceedings*.

Richard Socher, Eric H Huang, Jeffrey Pennington, Christopher D Manning, and Andrew Y Ng. 2011a. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211.

Richard Socher, Andrej Karpathy, Quoc V. Le, Christopher D. Manning, and Andrew Y. Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2:207–218.

Richard Socher, Cliff Chiung-Yu Lin, Andrew Y. Ng, and Christopher D. Manning. 2011b. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 129–136.

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 the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387*.

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 (Volume 1: Long Papers)*, pages 1556–1566.

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 *Advances in Neural Information Processing Systems*, pages 5998–6008.

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.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.