Tree-based Convolution for Sentence Modeling*

Mingbo Ma† **Liang Huang**† ‡ **Bing Xiang**‡ **Bowen Zhou**‡

†Graduate Center & Queens College
City University of New York
{mma2,lhuang}gc.cuny.edu

‡ IBM Watson Group
T. J. Watson Research Center, IBM
{lhuang,bingxia,zhou}@us.ibm.com

Abstract

In sentence modeling and classification, convolutional neural network approaches have recently achieved state-of-the-art results, but all such efforts process word vectors sequentially and neglect long-distance dependencies. To exploit both deep learning and linguistic structures, we propose a tree-based convolutional neural network model which exploit various long-distance relationships between words. Our model improves the sequential baselines on all three sentiment and question classification tasks, and achieves the highest published accuracy on TREC.

1 Introduction

Convolutional neural networks (CNNs), originally invented in computer vision (LeCun et al., 1995), has recently attracted much attention in natural language processing (NLP) on problems such as sequence labeling (Collobert et al., 2011), semantic parsing (Yih et al., 2014), and search query retrieval (Shen et al., 2014). In particular, recent work on CNN-based sentence modeling (Kalchbrenner et al., 2014; Kim, 2014) has achieved excellent, often state-of-the-art, results on various classification tasks such as sentiment, subjectivity, and question-type classification. However, despite their celebrated success, there remains a major limitation from the linguistics perspective: CNNs, being invented on pixel matrices in image processing, only consider sequential n-grams that are consecutive on the surface string and neglect long-distance dependencies, while the latter play an important role in many linguistic phenomena such as negation, subordination, and *wh*-extraction, all of which might dully affect the sentiment, subjectivity, or other categorization of the sentence.

Indeed, in the sentiment analysis literature, researchers have incorporated long-distance information from syntactic parse trees, but the results are somewhat inconsistent: some reported small improvements (Gamon, 2004; Matsumoto et al., 2005), while some otherwise (Dave et al., 2003; Kudo and Matsumoto, 2004). As a result, syntactic features have yet to become popular in the sentiment analysis community. We suspect one of the reasons for this is data sparsity (according to our experiments, tree n-grams are significantly sparser than surface n-grams), but this problem has largely been alleviated by the recent advances in word embedding. Can we combine the advantages of both worlds?

So we propose a very simple tree-based convolutional neural networks. We show dependency tree-based convolutional neural networks (DTCNNs) in this paper, but it also could be easily extended to constituency tree-based convolutional neural networks (CTCNNs). Our model is similar to Kim (2014), but while his sequential CNNs put a word in its sequential context, ours considers a word and its parent, grand-parent, great-grand-parent, and siblings on the dependency tree. This way we incorporate long-distance information that are otherwise unavailable on the surface string.

Experiments on three classification tasks demonstrate the superior performance of our DTCNNs over the baseline sequential CNNs. In particular, our accuracy on the TREC dataset outperforms all previously published results in the literature, including those with heavy hand-engineered features.

2 Dependency Tree-based Convolution

The original CNN, first proposed by LeCun et al. (1995), applies convolution kernels on a series of continuous areas of given images, and was adapted to NLP by Collobert et al. (2011). Following Kim (2014), one dimensional convolution operates the convolution kernel in sequential order in Equation 1, where $\mathbf{x}_i \in \mathbb{R}^d$ represents the d di-

* This work was done at both IBM and CUNY, and was supported in part by DARPA FA8750-13-2-0041 (DEFT), and NSF IIS-1449278. We thank Yoon Kim for sharing his code, and James Cross and Kai Zhao for discussions.

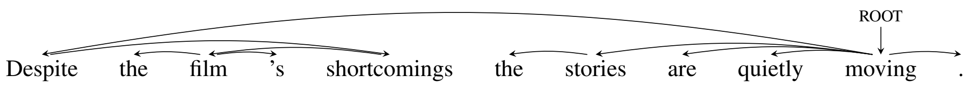


Figure 1: Running example from Movie Reviews dataset.

mensional word representation for the i-th word in the sentence, and $\oplus$ is the concatenation operator. Therefore $\widetilde{\mathbf{x}}_{i,j}$ refers to concatenated word vector from the i-th word to the $(i+j)$-th word:

$$\widetilde{\mathbf{x}}_{i,j} = \mathbf{x}_i \oplus \mathbf{x}_{i+1} \oplus \cdots \oplus \mathbf{x}_{i+j} \tag{1}$$

Sequential word concatenation $\widetilde{\mathbf{x}}_{i,j}$ works as n-gram models which feeds local information into convolution operations. However, this setting can not capture long-distance relationships unless we enlarge the window indefinitely which would inevitably cause the data sparsity problem.

In order to capture the long-distance dependencies we propose the dependency tree-based convolution model (DTCNN). Figure 1 illustrates an example from the Movie Reviews (MR) dataset (Pang and Lee, 2005). The sentiment of this sentence is obviously positive, but this is quite difficult for sequential CNNs because many n-gram windows would include the highly negative word "shortcomings", and the distance between "Despite" and "shortcomings" is quite long. DTCNN, however, could capture the tree-based bigram "Despite – shortcomings", thus flipping the sentiment, and the tree-based trigram "ROOT – moving – stories", which is highly positive.

2.1 Convolution on Ancestor Paths

We define our concatenation based on the dependency tree for a given modifier $\mathbf{x}_i$:

$$\mathbf{x}_{i,k} = \mathbf{x}_i \oplus \mathbf{x}_{p(i)} \oplus \cdots \oplus \mathbf{x}_{p^{k-1}(i)} \tag{2}$$

where function $p^k(i)$ returns the i-th word's k-th ancestor index, which is recursively defined as:

$$p^k(i) = \begin{cases} p(p^{k-1}(i)) & \text{if} \quad k > 0 \\ i & \text{if} \quad k = 0 \end{cases} \tag{3}$$

Figure 2 (left) illustrates ancestor paths patterns with various orders. We always start the convolution with x_i and concatenate with its ancestors. If the root node is reached, we add "ROOT" as dummy ancestors (vertical padding).

For a given tree-based concatenated word sequence $\mathbf{x}_{i,k}$, the convolution operation applies a filter $\mathbf{w} \in \mathbb{R}^{k \times d}$ to $\mathbf{x}_{i,k}$ with a bias term b described in equation 4:

$$c_i = f(\mathbf{w} \cdot \mathbf{x}_{i,k} + b) \tag{4}$$

where f is a non-linear activation function such as rectified linear unit (ReLu) or sigmoid function. The filter $\mathbf{w}$ is applied to each word in the sentence, generating the feature map $\mathbf{c} \in \mathbb{R}^l$:

$$\mathbf{c} = [c_1, c_2, \cdots, c_l] \tag{5}$$

where l is the length of the sentence.

2.2 Max-Over-Tree Pooling and Dropout

The filters convolve with different word concatenation in Eq. 4 can be regarded as pattern detection: only the most similar pattern between the words and the filter could return the maximum activation. In sequential CNNs, max-over-time pooling (Collobert et al., 2011; Kim, 2014) operates over the feature map to get the maximum activation $\hat{c} = \max \mathbf{c}$ representing the entire feature map. Our DTCNNs also pool the maximum activation from feature map to detect the strongest activation over the whole tree (i.e., over the whole sentence). Since the tree no longer defines a sequential "time" direction, we refer to our pooling as "max-over-tree" pooling.

In order to capture enough variations, we randomly initialize the set of filters to detect different structure patterns. Each filter's height is the number of words considered and the width is always equal to the dimensionality d of word representation. Each filter will be represented by only one feature after max-over-tree pooling. After a series of convolution with different filter with different heights, multiple features carry different structural information become the final representation of the input sentence. Then, this sentence representation is passed to a fully connected soft-max layer and outputs a distribution over different labels.

Neural networks often suffer from overtraining. Following Kim (2014), we employ random dropout on penultimate layer (Hinton et al., 2014). in order to prevent co-adaptation of hidden units. In our experiments, we set our drop out rate as 0.5 and learning rate as 0.95 by default. Following Kim (2014), training is done through stochastic gradient descent over shuffled mini-batches with the Adadelta update rule (Zeiler, 2012).

ancestor paths		siblings	
n	pattern(s)	n	pattern(s)
3	m h g	2	s m _
4	m h g g^2	3	s m h t s m _
5	m h g g^2 g^3	4	t s m h s m h g

Figure 2: Tree-based convolution patterns. Word concatenation always starts with m, while h, g, and g^2 denote parent, grand parent, and great-grand parent, etc., and "_" denotes words excluded in convolution.

2.3 Convolution on Siblings

Ancestor paths alone is not enough to capture many linguistic phenomena such as conjunction. Inspired by higher-order dependency parsing (McDonald and Pereira, 2006; Koo and Collins, 2010), we also incorporate siblings for a given word in various ways. See Figure 2 (right) for details.

2.4 Combined Model

Powerful as it is, structural information still does not fully cover sequential information. Also, parsing errors (which are common especially for informal text such as online reviews) directly affect DTCNN performance while sequential n-grams are always correctly observed. To best exploit both information, we want to combine both models. The easiest way of combination is to concatenate these representations together, then feed into fully connected soft-max neural networks. In these cases, combine with different feature from different type of sources could stabilize the performance. The final sentence representation is thus:

$$\hat{\mathbf{c}} = [\underbrace{\hat{c}_a^{(1)}, ..., \hat{c}_a^{(N_a)}}_{\text{ancestors}}; \underbrace{\hat{c}_s^{(1)}, ..., \hat{c}_s^{(N_s)}}_{\text{siblings}}; \underbrace{\hat{c}^{(1)}, ..., \hat{c}^{(N)}}_{\text{sequential}}]$$

where N_a, N_s, and N are the number of ancestor, sibling, and sequential filters. In practice, we use 100 filters for each template in Figure 2 . The fully combined representation is 1100-dimensional by contrast to 300-dimensional for sequential CNN.

3 Experiments

Table 1 summarizes results in the context of other high-performing efforts in the literature. We use three benchmark datasets in two categories: sentiment analysis on both Movie Review (MR) (Pang and Lee, 2005) and Stanford Sentiment Treebank (SST-1) (Socher et al., 2013) datasets, and question classification on TREC (Li and Roth, 2002).

For all datasets, we first obtain the dependency parse tree from Stanford parser (Manning et al., 2014).[1] Different window size for different choice of convolution are shown in Figure 2. For the dataset without a development set (MR), we randomly choose 10% of the training data to indicate early stopping. In order to have a fare comparison with baseline CNN, we also use 3 to 5 as our window size. Most of our results are generated by GPU due to its efficiency, however CPU potentially could generate better results.[2] Our implementation[3] is on top of Kim (2014)'s code.[4]

3.1 Sentiment Analysis

Both sentiment analysis datasets (MR and SST-1) are based on movie reviews. The differences between them are mainly in the different numbers of categories and whether the standard split is given. There are 10,662 sentences in the MR dataset. Each instance is labeled positive or negative, and in most cases contains one sentence. Since no standard data split is given, following the literature we use 10 fold cross validation to include every sentence in training and testing at least once. Concatenating with sibling and sequential information obviously improves tree-based CNNs, and the final model outperforms the baseline sequential CNNs by 0.4, and ties with Zhu et al. (2015).

Different from MR, the Stanford Sentiment Treebank (SST-1) annotates finer-grained labels, very positive, positive, neutral, negative and very negative, on an extension of the MR dataset. There are 11,855 sentences with standard split. Our model achieves an accuracy of 49.5 which is second only to Irsoy and Cardie (2014).

[1] The phrase-structure trees in SST-1 are actually automatically parsed, and thus can not be used as gold-standard trees.

[2] GPU only supports float32 while CPU supports float64.

[3] https://github.com/cosmmb/DTCNN

[4] https://github.com/yoonkim/CNN_sentence

Category	Model	MR	SST-1	TREC	TREC-2
This work	DTCNNs: ancestor	80.4†	47.7†	95.4†	88.4†
	DTCNNs: ancestor+sibling	81.7†	48.3†	**95.6†**	89.0†
	DTCNNs: ancestor+sibling+sequential	**81.9**	49.5	95.4†	88.8†
CNNs	CNNs-non-static (Kim, 2014) – baseline	81.5	48.0	93.6	86.4*
	CNNs-multichannel (Kim, 2014)	81.1	47.4	92.2	86.0*
	Deep CNNs (Kalchbrenner et al., 2014)	-	48.5	93.0	-
Recursive NNs	Recursive Autoencoder (Socher et al., 2011)	77.7	43.2	-	-
	Recursive Neural Tensor (Socher et al., 2013)	-	45.7	-	-
	Deep Recursive NNs (Irsoy and Cardie, 2014)	-	**49.8**	-	-
Recurrent NNs	LSTM on tree (Zhu et al., 2015)	**81.9**	48.0	-	-
Other deep learning	Paragraph-Vec (Le and Mikolov, 2014)	-	48.7	-	-
Hand-coded rules	SVM_S (Silva et al., 2011)	-		95.0	**90.8**

Table 1: Results on Movie Review (MR), Stanford Sentiment Treebank (SST-1), and TREC datasets. TREC-2 is TREC with fine grained labels. †Results generated by GPU (all others generated by CPU). *Results generated from Kim (2014)'s implementation.

3.2 Question Classification

In the TREC dataset, the entire dataset of 5,952 sentences are classified into the following 6 categories: abbreviation, entity, description, location and numeric. In this experiment, DTCNNs easily outperform any other methods even with ancestor convolution only. DTCNNs with sibling achieve the best performance in the published literature. DTCNNs combined with sibling and sequential information might suffer from overfitting on the training data based on our observation. One thing to note here is that our best result even exceeds SVM_S (Silva et al., 2011) with 60 hand-coded rules.

The TREC dataset also provides subcategories such as numeric:temperature, numeric:distance, and entity:vehicle. To make our task more realistic and challenging, we also test the proposed model with respect to the 50 subcategories. There are obvious improvements over sequential CNNs from the last column of Table 1. Like ours, Silva et al. (2011) is a tree-based system but it uses constituency trees compared to ours dependency trees. They report a higher fine-grained accuracy of 90.8 but their parser is trained *only* on the QuestionBank (Judge et al., 2006) while we used the standard Stanford parser trained on both the Penn Treebank and QuestionBank. Moreover, as mentioned above, their approach is rule-based while ours is automatically learned.

3.3 Discussions and Examples

Compared with sentiment analysis, the advantage of our proposed model is obviously more substantial on the TREC dataset. Based on our error analysis, we conclude that this is mainly due to the

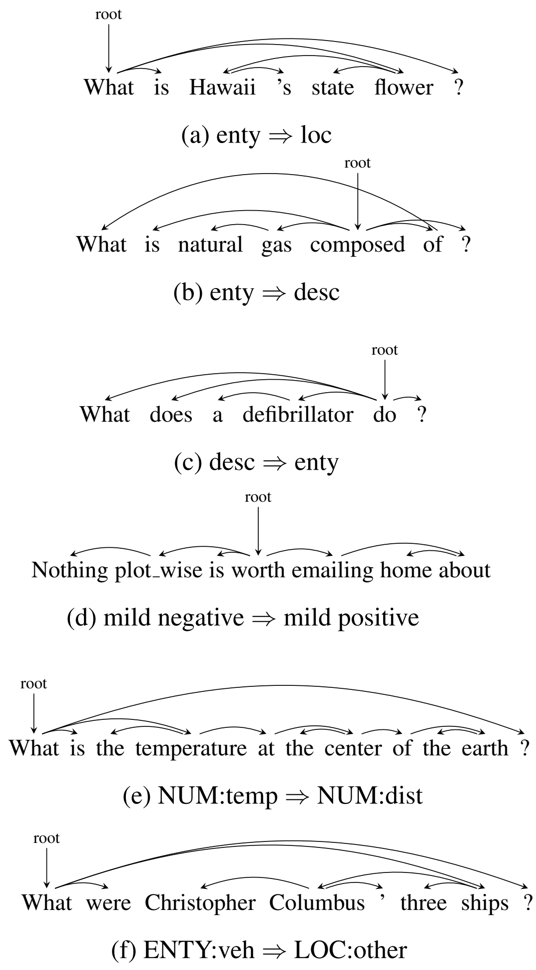
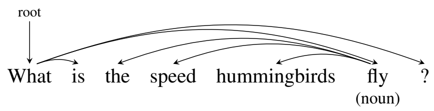
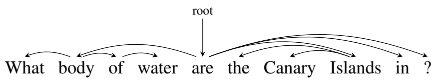


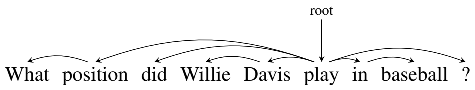
Figure 3: Examples from TREC (a–c), SST-1 (d) and TREC with fine-grained label (e–f) that are misclassified by the baseline CNN but correctly labeled by our DTCNN. For example, (a) should be *entity* but is labeled *location* by CNN.



(a) num ⇒ enty



(b) loc ⇒ enty



(c) hum ⇒ enty

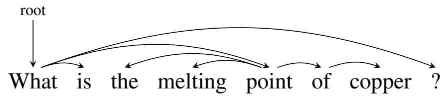
Figure 4: Examples from TREC datasets that are misclassified by DTCNN but correctly labeled by baseline CNN. For example, (a) should be *numerical* but is labeled *entity* by DTCNN.

difference of the parse tree quality between the two tasks. In sentiment analysis, the dataset is collected from the *Rotten Tomatoes* website which includes many irregular usage of language. Some of the sentences even come from languages other than English. The errors in parse trees inevitably affect the classification accuracy. However, the parser works substantially better on the TREC dataset since all questions are in formal written English, and the training set for Stanford parser[5] already includes the QuestionBank (Judge et al., 2006) which includes 2,000 TREC sentences.

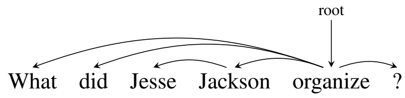
Figure 3 visualizes examples where CNN errs while DTCNN does not. For example, CNN labels (a) as *location* due to "Hawaii" and "state", while the long-distance backbone "What – flower" is clearly asking for an *entity*. Similarly, in (d), DTCNN captures the obviously negative tree-based trigram "Nothing – worth – emailing". Note that our model also works with non-projective dependency trees such as the one in (b). The last two examples in Figure 3 visualize cases where DTCNN outperforms the baseline CNNs in fine-grained TREC. In example (e), the word "temperature" is at second from the top and is root of a 8 word span "the ... earth". When we use a window of size 5 for tree convolution, every words in that span get convolved with "temperature" and this should be the reason why DTCNN get correct.

Figure 4 showcases examples where baseline CNNs get better results than DTCNNs. Example (a) is misclassified as *entity* by DTCNN due to parsing/tagging error (the Stanford parser performs its own part-of-speech tagging). The word "fly" at the end of the sentence should be a verb instead of noun, and "hummingbirds fly" should be a relative clause modifying "speed".

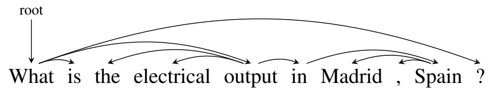
[5] http://nlp.stanford.edu/software/parser-faq.shtml



(a) num ⇒ enty and desc



(b) hum ⇒ enty and enty



(c) enty ⇒ num and num

Figure 5: Examples from TREC datasets that are misclassified by both DTCNN and baseline CNN. For example, (a) should be *numerical* but is labeled *entity* by DTCNN and *description* by CNN.

There are some sentences that are misclassified by both the baseline CNN and DTCNN. Figure 5 shows three such examples. Example (a) is not classified as *numerical* by both methods due to the ambiguous meaning of the word "point" which is difficult to capture by word embedding. This word can mean location, opinion, etc. Apparently, the numerical aspect is not captured by word embedding. Example (c) might be an annotation error.

We find DTCNNs' mistakes are mainly limited by two factors: the accuracy of parse tree and the quality of word embedding. Future work will focus on these two issues.

After we finish our work we found a parallel unpublished work (Mou et al., 2015) which proposes similar models. They use both constituency and dependency trees, but the dependency model always includes a node and all its children, which is a variant of our sibling model and always flat. By contrast, our ancestor model looks at the vertical path from any word to its ancestors, which is linguistically motivated (Shen et al., 2008).

4 Conclusions and Future Work

We have presented a very simple dependency tree-based convolution framework which outperforms sequential CNN baselines on various classification tasks. Extensions of this model would consider dependency labels and constituency trees. Also, we would evaluate on gold-standard parse trees.

References

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. Natural language processing (almost) from scratch. volume 12, pages 2493–2537.

Kushal Dave, Steve Lawrence, and David M Pennock. 2003. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of the 12th international conference on World Wide Web*, pages 519–528. ACM.

Michael Gamon. 2004. Sentiment classification on customer feedback data: noisy data, large feature vectors, and the role of linguistic analysis. In *Proceedings of COLING*, page 841.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Improving neural networks by preventing co-adaptation of feature detectors. *JMLR*, 15.

Ozan Irsoy and Claire Cardie. 2014. Deep recursive neural networks for compositionality in language. In *Advances in Neural Information Processing Systems*, pages 2096–2104.

John Judge, Aoife Cahill, and Josef van Genabith. 2006. Questionbank: Creating a corpus of parse-annotated questions. In *Proceedings of the 21st International Conference on Computational Linguistics*, pages 497–504.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 655–665.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of EMNLP*.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of ACL*, pages 1–11. Association for Computational Linguistics.

Taku Kudo and Yuji Matsumoto. 2004. A boosting algorithm for classification of semi-structured text. In *Proceedings of EMNLP*.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of ICML*.

Y. LeCun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Mller, E. Sckinger, P. Simard, and V. Vapnik. 1995. Comparison of learning algorithms for handwritten digit recognition. In *Int'l Conf. on Artificial Neural Nets*.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, pages 1–7. Association for Computational Linguistics.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of ACL: Demonstrations*, pages 55–60.

Shotaro Matsumoto, Hiroya Takamura, and Manabu Okumura. 2005. Sentiment classification using word sub-sequences and dependency sub-trees. In *Proceedings of PA-KDD*.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL*.

Lili Mou, Hao Peng, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2015. Discriminative neural sentence modeling by tree-based convolution. In *arXiv http://arxiv.org/abs/1504.01106v5*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of ACL*, pages 115–124.

Libin Shen, Lucas Champollion, and Aravind K Joshi. 2008. Ltag-spinal and the treebank. *Language Resources and Evaluation*, 42(1):1–19.

Yelong Shen, Xiaodong he, Jianfeng Gao, Li Deng, and Gregoire Mesnil. 2014. Learning semantic representations using convolutional neural networks for web search. WWW 2014, April.

J. Silva, L. Coheur, A. C. Mendes, and Andreas Wichert. 2011. From symbolic to sub-symbolic information in question classification. volume 35.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. In *Proceedings of EMNLP 2011*.

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 EMNLP 2013*.

Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 643–648.

Mattgew Zeiler. 2012. Adadelta: An adaptive learning rate method. arxiv: abs/1212.5701.

Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long short-term memory over tree structures. In *In Proceedings of ICML*.