Tree-based Convolution: A New Neural Architecture for Sentence Modeling

Lili Mou; Hao Peng; Ge Li; Yan Xu, Lu Zhang, Zhi Jin
{moull12, lige, xuyan14, zhanglu, zhijin}@sei.pku.edu.cn
penghao.pku@gmail.com
Software Institute, Peking University, 100871, P. R. China

Abstract

This paper proposes a new convolutional neural architecture based on tree-structures, called the tree-based convolutional neural network (TBCNN). Two variants take advantage of constituency trees and dependency trees, respectively, to model sentences. Compared with traditional “flat” convolutional neural networks (CNNs), TBCNNs explore explicitly sentences’ structural information; compared with recursive neural networks, TBCNNs have shorter propagation paths, enabling more effective feature learning and extraction. We evaluated our model in two widely applied benchmarks—sentiment analysis and question classification. Our models outperformed most state-of-the-art results, including both existing neural networks and dedicated feature/rule engineering.

1 Introduction

Recent years have witnessed an increasing trend in neural natural language processing (NLP). (Bengio et al., 2003; Mikolov et al., 2013) proposed unsupervised approaches to learn word embeddings, mapping discrete words to real-valued vectors and enabling neural networks to capture word meanings. (Collobert and Weston, 2008) proposed a unified convolutional model on sentences, applying it to a variety of tasks, e.g., part-of-speech tagging, named entity recognition, semantic role labeling, etc. (Socher et al., 2011) proposed a class of recursive neural networks, outperforming traditional machine learning algorithms by nearly 10% in a sentiment analysis task.

Sentence modeling, particularly important to various tasks, aims to capture sentence meanings. Existing neural sentence models mainly fall into two groups: convolutional neural networks (CNNs) and recursive neural networks (RNNs). In CNNs like (Collobert et al., 2011; Blunsom et al., 2014; Hu et al., 2014), a fixed-size window slides over time (successive words in sequence) to extract local features of a sentence; then they pool these features to a vector—usually taking the maximum value in each dimension—for supervised learning. Such models do not consider inherent sentence structures and fail to model semantically related words that fall apart.

Recursive neural networks (RNNs), on the other hand, take advantage of the sentences’ parsing trees (Socher et al., 2011; Socher et al., 2013a; Hermann and Blunsom, 2013). Each node in the tree is represented distributively as a vector; information is propagated recursively along the tree by some elaborate semantic composition (Blacoe and Lapata, 2012); finally, the root information is used for supervised learning. Although RNNs encode prior knowledge on tree structures into networks to some extent, they may have difficulty in learning deep dependencies because of long propagation paths (Erhan et al., 2009). The root node, which classification is completely based on, then becomes the bottleneck of such models.

To explore tree structural information effectively with short paths, we propose a novel neural architecture, called the Tree-based Convolutional Neural Network (TBCNN). Two variants are built upon constituency trees and dependency trees, denoted as c-TBCNN and d-TBCNN, respectively. Our models integrate convolution with tree structures. The idea is to apply a fixed-depth subtree feature detectors sliding over the entire parsing tree of a sentence, serving as the convolution kernel; then these features are pooled to one or more vectors for further processing. One virtue of our
model is that all features along the tree have short propagation paths to the output layer, and hence structural information can be learned effectively and efficiently by tree-based convolution.

Our models are evaluated in two benchmarks, sentiment analysis and question classification. We outperformed state-of-the-art results in both tasks. It is also the first time that neural networks, without utilizing human prior knowledge, outperforms dedicated feature/rule engineering in the question classification task.

2 Background and Related Work

In this section, we present the background and related work regarding neural architectures of sentence modeling.

2.1 Convolutional Neural Networks

Convolutional neural networks (CNNs), first proposed for image processing (LeCun et al., 1995), turn out to work with natural languages as well. Figure 1 (a) presents the architecture of the classical convolution process on a sentence (Collobert and Weston, 2008). A fixed window, called convolution kernel, slides over the sentence, and outputs the features extracted. Let $t$ be the window size, and $x_1, \ldots, x_t \in \mathbb{R}^n$ be $n$-dimensional column vectors corresponding to the words in the current window. The output of convolution, evaluated at the current position, is

$$y = f(W \cdot [x_1; \cdots; x_t] + b)$$

where $y \in \mathbb{R}^m$ ($m$ is the number of convolution kernels), $W \in \mathbb{R}^{m \times 2n}$, $b \in \mathbb{R}^m$ are parameters, and $f$ is the activation function. Semicolons represent column vector concatenation.

One layer of convolution can extract neighboring information effectively. But the features are “local,” because words that are not in one convolution window do not interact even though they may be semantically related. (Blunsom et al., 2014) builds deep convolutional models so that local features can mix at high-level layers. Similar deep CNN architectures include (Kim, 2014; Hu et al., 2014). All these models are “flat,” by which we mean no structural information is used explicitly. As evidence in the literature shows, CNNs are particularly suitable for noisy and short texts, such as twitter or microblogs (Kim, 2014).

2.2 Recursive Neural Networks

Recursive neural networks (RNNs) were proposed in (Socher et al., 2011) to model sentences. In the original version, each node in the tree structure has a distributed, real-valued representation. Let node $p$ be the parent of $c_1$ and $c_2$ in a constituency parsing tree, the vector of which are denoted as $p, c_1, c_2 \in \mathbb{R}^n$ ($n$ is the dimension of embeddings). Then the parent’s representation is composited by its children’s according to the following equation.

$$p = f(W[c_1; c_2] + b)$$

where $W \in \mathbb{R}^{n \times 2n}$ and $b \in \mathbb{R}^n$ are model parameters. This process, depicted in Figure 1 (b), is done recursively along the tree; the root vector is then used for supervised classification, e.g., sentiment analysis. Dependency trees and combinatory categorial grammar can also be used as RNN structures (Socher et al., 2013a; Hermann and Blunsom, 2013). Deep RNNs are built in (Irsoy and Cardie, 2014) to enhance information interaction. Improvement for semantic compositionality in RNNs include matrix-vector interaction (Socher et al., 2012), tensor interaction (Socher et al., 2013b), etc. These methods are more suit-
able of capturing the nature of sentences regarding logic expressions like exclamation, negation, etc.

One major drawback of RNNs is the long propagation path of information near leaf nodes. As gradient may vanish when propagated through a deep path, such long dependency buries illuminating information under a complicated neural architecture, leading to the difficulty of training. As reported in (Mou et al., 2014a), RNNs may not be effective for training very deep tree structures, such as program source code’s parsing trees. Long short term memory (LSTM), first proposed for modeling temporal data (Hochreiter and Schmidhuber, 1997), is integrated to RNNs to alleviate this problem (Tai et al., 2015; Le and Zuidema, 2015; Zhu et al., 2015).

A degraded variant of RNNs is the recurrent network (Bengio et al., 1994), whose architecture can be viewed as a right-most tree. As meaningful tree structures are not used, they are also particularly suitable for modeling short or noisy texts, similar to CNNs (Shang et al., 2015).

Different from the above approaches, our proposed c-TBCNN and d-TBCNN models utilize tree structures explicitly in a novel way by convolving a set of subtree feature detectors, which can be learned effectively because of short dependency paths. TBCNN was first proposed in our previous work for programming language processing (Mou et al., 2014b; Mou et al., 2014a). In this paper, we propose two variants, called c-TBCNN and d-TBCNN, extending tree-based convolution to natural languages.

3 Tree-based Convolution

In this section, we propose the novel tree-based convolutional neural network (TBCNN) for sentence modeling. Figure 2(c) depicts the tree convolution process. The dashed triangle represents a convolution kernel, extracting structural features along the tree. Then, the extracted features are packed into one or more fixed-size vectors by max-pooling, allowing short propagation path between the output layer and any position in the tree. The tree-based convolution, along with the pooling method, allows effective structural feature learning.

In the rest of this section, we describe two variants of TBCNN, built up on constituency trees (Section 3.1) and dependency trees (Section 3.2). Several heuristics are introduced in Section 3.3, the training objective is presented in Section 3.4.

3.1 c-TBCNN

Figure 2 illustrates an example of the constituency tree, corresponding to the sentence “I like it.” (See the blue nodes on the left-hand side.) Leaf nodes in a constituency tree are words in the sentence; non-leaf nodes represent a grammatical constituent, e.g., a noun phrase, a verbal phrase, etc. In our experiment, sentences are parsed by the Stanford parser; further, constituency trees are binarized for simplicity.

To process constituency trees effectively by convolution, both leaf nodes and non-leaf nodes should be represented as real-valued vectors in the same semantic space. In our c-TBCNN model, leaf nodes have pretrained $n_e$-dimensional embeddings by unsupervised algorithms (Mikolov et al., 2013); non-leaf nodes are coded by Equation 1 similar to the RNN in (Socher et al., 2011).

We now consider a constituency tree-based convolution with two-layer kernels. If we have $n_e$ feature detectors for convolution, the output—evaluated at a local position of subtree $p \leftarrow (c_1, c_2)$ with vector representations $p$, $c_1$, and $c_2$—is

$$y = f \left( W_p^{(c)} \cdot p + W_{l}^{(c)} \cdot c_1 + W_r^{(c)} \cdot c_2 + b^{(c)} \right)$$

where $y \in \mathbb{R}^{n_e}$; $b \in \mathbb{R}^{n_e}$ is the bias term. $W_p^{(c)}$, $W_{l}^{(c)}$, $W_r^{(c)} \in \mathbb{R}^{n_e \times n_e}$ are weights associated with parent $p$, left child $c_1$, and right child $c_2$, respectively. The superscript $(c)$ indicates the weights are for c-TBCNN models. For leaf nodes that do not have enough depth for the convolution window, we set $c_1$ and $c_2$ to be 0. Like other convolution processes, such as (Collobert and Weston, 2008), weights $W_p^{(c)}$, $W_{l}^{(c)}$, $W_r^{(c)}$ are shared among different positions along the tree structure.

Tree-based convolution kernels can be extended to arbitrary numbers of layers straightforwardly. The complexity is exponential with respect to the depth of kernels, but it is linear to the number of nodes we convolve in a window. Hence, tree-based convolution, compared with “flat” CNNs, does not add to computational cost, provided the same amount of information to process at a time. In our experiment, we use 2-layer kernels.

\[\text{http://nlp.stanford.edu/software/lex-parser.shtml} \]

\[\text{http://nlp.stanford.edu/software/lex-parser.shtml} \]

\[\text{http://nlp.stanford.edu/software/lex-parser.shtml} \]
3.2 d-TBCNN

Dependency trees are another representation of sentence structures. Each node in dependency trees corresponds to a word; an edge \(a \rightarrow b\) indicates \(a\) is governed by \(b\) in Figure 2(b). Edges are labeled with grammatical relation types by the parser, e.g., nsubj, (de Marneffe et al., 2006).

In our d-TBCNN model, we also apply two-layer convolution kernels to extract structural features. However, in d-TBCNN, different nodes may have different numbers of children. We assign a weight matrix \(W_r\) to each relation type \(r\). Rare relation types (less than 3,000 occurrences) are mapped to one shared weight matrix. Let \(p \leftarrow (c_1, \cdots, c_n)\) be the words and relations in the convolution kernel evaluated at a certain position. The output of the feature detectors is

\[
y = f \left( W_p^{(d)} \cdot p + \sum_{i=1}^{n} W_r^{(d)} \cdot c_i + b^{(d)} \right)
\]

where \(y \in \mathbb{R}^{ne}; b^{(d)} \in \mathbb{R}^{ne}\) is the bias term. \(W_p^{(d)} \in \mathbb{R}^{ne \times ne}\) is the weight parameter for parent; \(W_r^{(d)} \in \mathbb{R}^{ne \times ne}\) is the weight for child \(c_i\), whose grammatical relation is \(r[c_i]\). Superscript \(d\) indicates the weights are for d-TBCNN.

Both c-TBCNN and d-TBCNN have their own advantages: d-TBCNN exploits structural features more efficiently because of the compact expressiveness of dependency trees; c-TBCNN, on the other hand, is more effective in integrating global features due to the underneath coding process.

3.3 Pooling Heuristics

As different sentences may have different lengths and tree structures, the extracted features by tree-based convolution also vary in size and shape. Dynamic pooling is a prevailing technique for dealing with this problem. We present several heuristics for pooling in both c-TBCNN and d-TBCNN.

- **Global pooling.** All features are pooled to one vector, and we take the maximum value in each dimension. This simple heuristic is applicable to any structure, including c-TBCNN and d-TBCNN.

- **3-way pooling for c-TBCNN.** To preserve more information over different regions of constituency trees, we propose 3-way pooling. If a tree has maximum depth \(d\), we pool nodes of less than \(\alpha \cdot d\) layers to a TOP vector (\(\alpha\) is set to 0.6 in our experiment); lower nodes are pooled to LOWER_LEFT or LOWER_RIGHT according to their relative position with respect to the root node.

- **k-way pooling for d-TBCNN.** Nodes in dependency trees are one-one corresponding to words in a sentence. Thus, a total order on features extracted by convolution can be defined by their corresponding word order. If we would like to pool the features to \(k\) vectors, with indexes \(1, 2, \cdots, k\), we can adopt an “equal allocation” strategy. Let \(i\) be the position of a word in a sentence \((i = 1, 2, \cdots, n)\). Its feature is pooled to \(j\)-th vec-
| Task                  | Data samples                                                                 | Label |
|----------------------|------------------------------------------------------------------------------|-------|
| Sentiment Analysis   | Offers that rare combination of entertainment and education. ++               |       |
|                      | An idealistic love story that brings out the latent 15-year-old romantic in everyone. + |       |
|                      | Its mysteries are transparently obvious, and it’s too slowly paced to be a thriller. - |       |
| Question Classification | What is the temperature at the center of the earth? NUMBER                   |       |
|                      | What state did the Battle of Bighorn take place in? LOCATION                 |       |

$$tor, \text{ if } (j - 1) \frac{n}{k} \leq i \leq j \frac{n}{k}$$

3.4 Training the Networks

After pooling, the information is packed into one or more fixed-size vectors, which can be fed forward to a fully-connected hidden layer. Then we apply a softmax layer, predicting the probability of each label, i.e., $p(y|x)$ satisfying $p(y|x)$ satisfies $\sum_{i=1}^{c} p(y_i|x) = 1$, where $x$ refers to an input sample and $c$ is the number of classes to predict.

The cost function of a sample is the standard cross entropy error

$$J = \sum_{i=1}^{c} t_i \log y_i$$

where $t \in \mathbb{R}^c$ is the ground truth, with the target label represented as a one-hot vector.

We do not apply $\ell_2$ regularization for weights or embeddings temporarily. Instead, dropout is used. Gradients are computed by standard backpropagation; stochastic gradient descent with momentum of mini-batch version is used for optimization.

4 Experimental Results

In this section, we present two experiments, sentiment analysis and question classification.

4.1 Sentiment Analysis

4.1.1 The Task and Dataset

Sentiment analysis is a widely-used benchmark for sentence modeling. The dataset\(^3\) consists of more than 10,000 movie reviews; the task is to predict reviews’ sentiment among 5 target labels (strongly positive, positive, neutral, negative, and strongly negative). Some examples are shown in Table 1. We use the standard split for training, validating and testing, containing 8544, 1101, 2210 sentences, respectively. In the training set, each sentence and phrase is tagged with its sentiment label. We regard phrases as individual samples, and hence the training set has more than 15k samples in total. For validating and testing, we only consider the sentiment of complete sentences (root labels). Both c-TBCNN and d-TBCNN use the Stanford parser; a few sentences (or phrases) parsed abnormally are discarded (less than 1%).

4.1.2 Hyperparameter Tuning

To train our model, we applied mini-batch version of stochastic gradient descent with momentum (0.9); the batch size is 500. We tried AdaGrad (Duchi et al., 2011), which turns out to be more stable but slightly worse than high-variance algorithms.

We tried \(\text{tanh}\) and \(\text{ReLU}\) as our activation function, and found that \(\text{ReLU}\) helps optimization, achieving high accuracy in both training and validation sets.

We tried three settings of dropout to regularize our model: (1) dropout the last hidden layer (2) dropout all hidden layers (3) further dropout embeddings. We found that dropout helps prevent overfitting to a large extent in terms of long-term performance during training—the decrease of validation accuracy due to overfitting is much smaller than non-dropout networks. In this experiment, all hidden layers are dropped out by 50% and embeddings are dropped out by 40%.

The number of hidden units in this experiment is 300 for convolution and 200 for the last hidden layer; word embeddings are 300 dimensional. We found our models yield similar performance provided the dimension is large (200 vis-a-vis 300), but they are better than 50 dimensional.

For detailed configurations, please refer to our website\(^4\).

4.1.3 Performance

Table 2 compares our models to state-of-the-art results in the task of sentiment analysis. As we see,
Table 2: Accuracy of 5-class sentiment prediction (in percentage).

| Group      | Method                  | 5-class accuracy | Reported in                  |
|------------|-------------------------|------------------|------------------------------|
| Baseline   | SVM                     | 40.7             | (Socher et al., 2013b)       |
|            | Naïve Bayes             | 41.0             | (Socher et al., 2013b)       |
| CNNs       | 1-layer convolution     | 37.4             | (Blunsom et al., 2014)       |
|            | Deep CNN                | 48.5             | (Blunsom et al., 2014)       |
| RNNs       | Perception-like         | 43.2             | (Socher et al., 2013b)       |
|            | Matrix-vector           | 44.4             | (Socher et al., 2013b)       |
|            | Tensor                  | 45.7             | (Socher et al., 2013b)       |
|            | LSTM                    | 48.0             | (Zhu et al., 2015)           |
|            | LSTM                    | 49.9             | (Le and Zuidema, 2015)       |
|            | LSTM                    | 50.6             | (Tai et al., 2015)           |
|            | Deep RNN                | 49.8             | (Irsoy and Cardie, 2014)     |
| Miscellaneous | Vector avg.          | 32.7             | (Socher et al., 2013b)       |
|            | Paragraph vector        | 48.7             | (Le and Mikolov, 2014)       |
| TBCNNs     | c-TBCNN                 | 49.1             | Our implementation           |
|            | d-TBCNN                 | 50.7             | Our implementation           |

d-TBCNN achieved 50.7% accuracy, outperforming the most state-of-the-art results achieved by long-short term memory based RNNs (Tai et al., 2015). c-TBCNN is a slightly worse, achieving 49.1% accuracy.

In a more controlled setting, with perception-like interaction and shallow architectures, TBCNNs, of both variants, consistently outperform RNNs to a large extent by 6–7%; they also consistently outperform “flat” CNNs by more than 10%. Such results suggest that structures are important when modeling sentences; tree-based convolution can capture these structural information more effectively than RNNs.

Further, as our current version of TBCNNs contains only one convolutional layer with simple perception-like interaction, our models are also open for improvement by designing more elaborate interactions (e.g., bilinear, tensor) and stack of deep architectures.

### 4.2 Question Classification

We further evaluated TBCNN models by a question classification task. The dataset contains 5452 annotated sentences plus 500 test samples in TREC 10. We also used standard split like Silva et al., 2011. The target labels contain 6 classes, namely abbreviation, entity, description, human, location, and numeric. Some examples are also shown in Table 1.

We chose this task to evaluate our model because the number of training samples is rather small, so that we can know TBCNN’s performance when it is applied to datasets of different sizes. To alleviate the sparseness of training data, we set the dimensions of convolutional and hidden layers to 30 and 25, respectively. Further, we do not backpropagate gradient to embeddings, which are dropout at rate 0.3; hidden layers’ dropout rate is 0.05.

Table 3 compares our model to various other methods. As we see, d-TBCNN is better than existing neural architectures by 2–5%, including CNNs, RNNs, and newly-proposed adaptive sentence models based on deep CNN. Also, our model outperforms former state-of-the-art result.
achieved by SVM with more than 10k features and 60 hand-coded rules (Silva et al., 2011). To our best knowledge, this is the first time that neural networks beat excessive human engineering in this task.

4.3 Conclusion and Future Work

In this paper, we propose a novel tree-based convolutional architecture (TBCNN) for sentence modeling. The model can be built upon either constituency trees or dependency trees.

We outperformed state-of-the-art results in two benchmarks, sentiment analysis and question classification. The results show that tree-based convolution can make use of tree structures effectively and efficiently.

As this paper first introduces the new TBCNN model, we are empowering the model by designing more meaningful interactions, e.g., bilinear or tensor interaction to model exclamations or negations. We are also trying to stack multiple layers of tree convolution to improve information integration.

References

[Bengio et al.1994] Yoshua Bengio, P. Simard, and P. Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks, 5(2):157–166.

[Bengio et al.2003] Yoshua Bengio, R. Ducharme, P. Vincent, and C. Jauvin. 2003. A neural probabilistic language model. Journal of Machine Learning Research, 3:1137–1155.

[Blacoe and Lapata2012] William Blacoe and M. Lapata. 2012. A comparison of vector-based representations for semantic composition. In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning.

[Blunsom et al.2014] Phil Blunsom, E. Grefenstette, and N. Kalchbrenner. 2014. A convolutional neural network for modelling sentences. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics.

[Collobert and Weston2008] Ronan Collobert and J. Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In Proceedings of the 25th International Conference on Machine learning.

[Collobert et al.2011] Ronan Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. Natural language processing (almost) from scratch. The Journal of Machine Learning Research, 12:2493–2537.

[de Marneffe et al.2006] Marie-Catherine de Marneffe, B. MacCartney, and C. Manning. 2006. Generating typed dependency parses from phrase structure parses. In Proceedings of Language Resource and Evaluation Conference, number 2006, pages 449–454.

[Duchi et al.2011] John Duchi, E. Hazan, and Y. Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. The Journal of Machine Learning Research, 12:2121–2159.

[Erhan et al.2009] Dumitru Erhan, Y. Bengio, S. Bengio, and P. Vincent. 2009. The difficulty of training deep architectures and the effect of unsupervised pre-training. In Proceedings of International Conference on Artificial Intelligence and Statistics.

[Hermann and Blunsom2013] Karl Moritz Hermann and P. Blunsom. 2013. The role of syntax in vector space models of compositional semantics. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics.

[Hochreiter and Schmidhuber1997] Sepp Hochreiter and J. Schmidhuber. 1997. Long short-term memory. Neural computation, 9(8):1735–1780.

[Hu et al.2014] Baotian Hu, D. Lu, H. Li, and Q. Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In Advances in Neural Information Processing Systems.

[Irsoy and Cardie2014] Ozan Irsoy and C. Cardie. 2014. Deep recursive neural networks for compositionality in language. In Advances in Neural Information Processing Systems.

[Kim2014] Yoon Kim. 2014. Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882.

[Le and Mikolov2014] Quoc Le and T. Mikolov. 2014. Distributed representations of sentences and documents. In Proceedings of the 31st International Conference on Machine Learning.

[Le and Zuidema2015] Phong Le and W. Zuidema. 2015. Compositional distributional semantics with long short term memory. arXiv preprint arXiv:1503.02510.

[LeCun et al.1995] Yann LeCun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Muller, and E. Sackinger. 1995. Comparison of learning algorithms for handwritten digit recognition. In Proceedings of International Conference on Artificial Neural Networks.
[Mikolov et al.2013] Tomas Mikolov, K. Chen, G. Corrado, and J. Dean. 2013. Efficient estimation of word representations in vector space. In ICLR Workshop.

[Mou et al.2014a] Lili Mou, G. Li, Z. Jin, L. Zhang, and T. Wang. 2014a. TBCNN: A tree-based convolutional neural network for programming language processing. arXiv preprint arXiv:1409.5718.

[Mou et al.2014b] Lili Mou, G. Li, X. Liu, H. Peng, Z. Jin, Y. Xu, and L. Zhang. 2014b. Building program vector representations for deep learning. arXiv preprint arXiv:1409.3358.

[Shang et al.2015] Lifeng Shang, D. Lu, and H. Li. 2015. Neural responding machine for short-text conversation. arXiv preprint arXiv:1503.02364.

[Silva et al.2011] Joao Silva, Luisa C., A. Cristina Mendes, and Andreas W. 2011. From symbolic to sub-symbolic information in question classification. Artificial Intelligence Review, 35(2):137–154.

[Socher et al.2011] Richard Socher, J. Pennington, E. Huang, A. Ng, and C. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In Proceedings of the Conference on Empirical Methods in Natural Language Processing.

[Socher et al.2012] Richard Socher, B. Huval, C. Manning, and A. 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.

[Socher et al.2013a] Richard Socher, Q. Le, C. Manning, and A. Ng. 2013a. Grounded compositional semantics for finding and describing images with sentences. In NIPS Deep Learning Workshop.

[Socher et al.2013b] Richard Socher, A. Perelygin, J. Wu, J. Chuang, C. Manning, A. Ng, and C. Potts. 2013b. Recursive deep models for semantic compositionality over a sentiment treebank. In Proceedings of Conference on Empirical Methods in Natural Language Processing.

[Tai et al.2015] Kai Sheng Tai, R. Socher, and D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. arXiv preprint arXiv:1503.00075.

[Zhao et al.2015] Han Zhao, Z. Lu, and P. Poupart. 2015. Self-adaptive hierarchical sentence model.

[Zhu et al.2015] Xiaodan Zhu, P. Sobhani, and Y. Guo. 2015. Long short-term memory over tree structures. arXiv preprint arXiv:1503.04881.