Attention-based Recursive Neural Networks for Semantic Representation

Yao Zhou, Cong Liu and **Yan Pan**
School of Data and Computer Science, Sun Yat-sen University
{zhouyao@mail2, liucong3@mail, panyan5@mail}.sysu.edu.cn

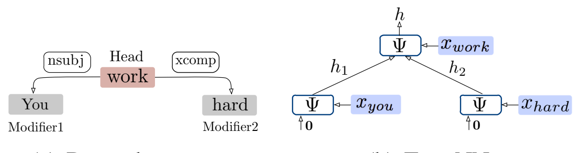
Abstract

The attention mechanism has succeeded in many NLP tasks. While existing attention-based models exert attention on sequential memories, we attempt to focus attention recursively on local tree structures. Specially, we use the attention mechanism to select semantically more relevant modifier words to construct a sentence representation on a dependency tree using LSTM and GRU, respectively. Our attention-based recursive neural networks outperform the non-attention counterparts in semantic relatedness and sentiment classification, and achieve the state-of-the-art performance.

1 Introduction

It has been shown that distributed representation of word achieves superior performance in many high-level NLP tasks. However, with word vectors, it is still challenging to obtain a good sentence representation due to the diversity in the sentence length and word order. There are two typical approaches for sentence-level semantic representations: sequence-based and tree-based models. The sequence-based models, such as the *recurrent neural networks* (RNNs) (Elman, 1990; Mikolov, 2012), treat composing a sentence representation as a sequential composition of words from left to right. On the other hand, the tree-based models, such as the *recursive neural networks* (Goller and Kuchler, 1996; Socher et al., 2011), compose a representation button up based on the parse tree of the sentence.

Amongst the tree-based neural networks (Tree-NNs), the Tree-LSTMs (Tai et al., 2015) recursively applies the LSTM to derive the representations for the internal nodes in the parse trees. Specifically, they propose an *N-ary Tree-LSTM* for constituent trees as well as a *Child-Sum Tree-LSTM* for dependency trees. Experiment results show that the Tree-LSTMs are effective for semantic representations.



(a) Dependency tree (b) Tree-NN

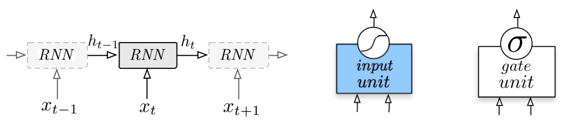
Figure 1: A tree-based neural network (Tree-NN), e.g. tree-LSTM, derives a sentence representation h using the word vector x_{work} and the recursive input h_1 and h_2 from the children based on the dependency tree.

In this paper, we extend the Child-Sum Tree-LSTM by using the *attention mechanism*. As shown in Figure 1(b), a Tree-NN derives the representation of each head word in the dependency tree from the representations of the head word's modifier words. The attention mechanism enables our model to derive better sentence representation by attaching difference emphases to the recursive input from the modifier words in forms of weights, which are obtained from a trained neural network. Our contributions are summarized as follows:

1. We firstly apply the attention mechanisms to the recursive Tree-NNs, and propose extensions to both GRU and LSTM.

2. We conduct extensive experiments on two semantic tasks: semantic relatedness (SemEval 2014, Task 1) and sentiment classification (Stanford Sentiment Treebank).

3. Our attention-based Tree-NNs outperform the non-attention counterparts and achieve the state-of-the-art performance.



(a) RNNs (b) Input and gate units

Figure 2: RNNs, input unit and gate unit in LSTM.

2 Preliminaries

In this section, we will briefly discuss RNN, LSTM, GRU, Tree-LSTM, as well as Tree-GRU, a simple variant of the Tree-LSTM.

2.1 Recurrent neural network (RNN)

Given an input sequence of T words, an RNN (Figure 2(a)) iteratively computes an output h_t using word x_t $(1 \leq t \leq T)$ and the RNN's previous output h_{t-1}. The output h_T for the last word in the sentence will be used as the representation of the whole sentence. Unfortunately, in such an RNN, the propagation of the memory of earlier part of the sentence diminishes in long sentences.

2.2 Long short-term memory (LSTM)

The LSTM (Hochreiter and Schmidhuber, 1997; Zaremba and Sutskever, 2014) preserves long-term memory by adding a recurrent memory cell. As shown in Figure 3(a), an LSTM has a memory cell c_t and three gates: the input gate i_t, the forget gate f_t and the output gate o_t at a time step t. The input u_t is computed by the *input unit* I, and the three gates, i_t, f_t, and o_t, are all computed by the same kind of *gate unit* G but with different parameter $\theta = (W, U, b)$. The input unit and the gate unit are shown in Figure 2(b) and defined in Equations 1.

$$\begin{aligned} I(x,h) &= \tanh(W^{(u)}x + U^{(u)}h + b^{(u)}), \\ G_\theta(x,h) &= \sigma(Wx + Uh + b). \end{aligned} \tag{1}$$

The LSTM is illustrated in Figure 3(a) and formally defined in Equations 2.

$$\begin{aligned} u_t &= I(x_t, h_{t-1}), \qquad f_t = G_{\theta_f}(x_t, h_{t-1}), \\ i_t &= G_{\theta_i}(x_t, h_{t-1}), \quad o_t = G_{\theta_o}(x_t, h_{t-1}), \\ c_t &= i_t \odot u_t + f_t \odot c_{t-1}, \\ h_t &= LSTM(x_t, h_{t-1}, c_{t-1}) \\ &= o_t \odot \tanh(c_t). \end{aligned} \tag{2}$$

2.3 Gated recurrent unit (GRU)

The GRU (Chung et al., 2014) is one of the simplified variants of the LSTM that has only two gates and the memory cell removed. As shown in Figure 3(d) and defined in Equations 3, in a GRU, the reset get r_t is similar to the f_t in LSTM, and the update gate z_t is similar to i_t and o_t in LSTM.

$$\begin{aligned} r_t &= G_{\theta_r}(x_t, h_{t-1}), \quad z_t = G_{\theta_z}(x_t, h_{t-1}), \\ \tilde{h}_t &= I(x_t, r_t \odot h_{t-1}), \\ h_t &= GRU(x_t, h_{t-1}) \\ &= z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t. \end{aligned} \tag{3}$$

2.4 Child-Sum Tree-LSTM

While conventional LSTM has a single pair of recurrent input h_t and c_t, being a Tree-NN (Figure 1(b)), the *Tree-LSTM* has multiple recursive input, $h_1, h_2, \ldots, h_k, c_1, c_2, \ldots, c_k$, where k is the number of children of the current node. The *Child-Sum Tree-LSTM* (Tai et al., 2015), as illustrated in Figure 3(b) and defined in Equations 4, handles multiple recursive input h_i by summing them, besides using multiple forget gates f_i for the recursive memory cells c_i. For clarity, in all of the Tree-NNs described thereafter, we omit the subscript for the current node.

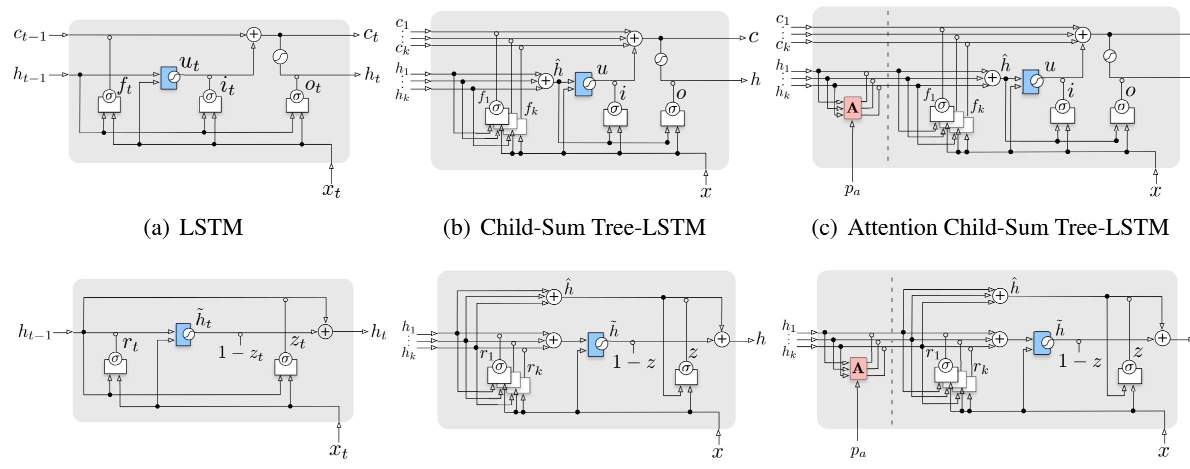
$$\begin{aligned} h &= LSTM^{tree}_{sum}(x, h_1 \ldots h_k, c_1 \ldots c_k) \\ &= LSTM(x, \hat{h}, \{c_1 \ldots c_k\}), \\ &\text{where:} \\ &\hat{h} = \sum_{1 \leq i \leq k} h_i, \quad f_i = G_{\theta_{f,i}}(x, h_i), \\ &\hat{c} = i \odot u + \sum_{1 \leq i \leq k} f_i \odot c_i. \end{aligned} \tag{4}$$

The *Child-Sum Tree-LSTM* is well suited for dependency trees, where the children (modifiers) are order-insensitive.

2.5 Child-Sum Tree-GRU

Similar to the way that *Child-Sum Tree-LSTM* extends the conventional LSTM in dependency trees, we define a *Child-Sum Tree-GRU*. As illustrated in Figure 3(e) and defined in Equations 5, the *Child-Sum Tree-GRU* defines multiple reset gates r_i for each recursive input h_i.

$$\begin{aligned} h &= GRU^{tree}_{sum}(x, h_1 \ldots h_k) = GRU(x, \hat{h}), \\ &\text{where:} \\ &\hat{h} = \sum_{1 \leq i \leq k} h_i, \quad r_i = G_{\theta_{r,i}}(x, h_i), \\ &\tilde{h} = I(x, \sum_{1 \leq i \leq k} r_i \odot h_i). \end{aligned} \tag{5}$$



(a) LSTM (b) Child-Sum Tree-LSTM (c) Attention Child-Sum Tree-LSTM

(d) GRU (e) Child-Sum Tree-GRU (f) Attention Child-Sum Tree-GRU

Figure 3: Illustrations of different architectures.

3 Attention-based Tree-NNs

3.1 The attention unit

Our attention mechanism for Tree-NNs is implemented by an *attention unit* A, which is defined below in Equations 6. Given the memory input from k children nodes, $h_1^{(in)}, h_2^{(in)}, \dots, h_k^{(in)}$, the attention unit A first computes a weight vector $w \in \mathbb{R}^k$ using a two-layer neural network, and then multiply each input $h_i^{(in)}$ with the weight w_i to produce each of its k output $h_i^{(out)}$.

$$
\begin{aligned}
&\alpha_i = V_i^{(a)} \tanh(W^{(a)} h_i^{(in)} + U^{(a)} p_a), \\
&w = \textit{softmax}([\alpha_1, \alpha_2, \dots, \alpha_k]), \\
&h_i^{(out)} = w_i h_i^{(in)}, \qquad (1 \le i \le k).
\end{aligned} \tag{6}
$$

In Equations 6, $V^{(a)}$, $W^{(a)}$ and $U^{(a)}$ are the hyper-parameters in the two-layer neural network; p_a is called the *attention parameter*, which is used to affect the weight w_i that is computed for each memory input $h_i^{(in)}$. We will discuss different implementations of p_a in Section 3.3.

3.2 Attention-based Tree-LSTM and GRU

Our *attention-based Tree-LSTM* (Figure 3(c)) and *attention-based Tree-GRU* (Figure 3(f)) are implemented by appending the Tree-LSTM and Tree-GRU to the *attention unit*, respectively. Concretely, the memory input $h_1, h_2, \dots, h_k$ from the children are weighted by the attention unit before entering a Tree-LSTM or Tree-GRU. Thus, different emphases are placed to different children inputs by the attention unit via the attention parameter.

For space limitations, this paper only proposed the attention-based extension to the Child-Sum Tree-LSTM and GRU. With the attention unit, it is straightforward to extend other RNNs, such as the *N-ary Tree-LSTM* (Tai et al., 2015).

3.3 Variants of the attention Tree-NN

The two attention-based Tree-NNs adopt different *attention parameter* implementations.

Static-attention Tree-NNs. In this implementation, the value of the *attention parameter* p_a is simply fixed to a constant. Since the weight w is computed by the attention unit, the value of p_a can be a small floating number.

Dynamic-attention Tree-NNs. This implementation uses the sentence representation from a simpler model (i.e. a sequential RNN) as p_a, and then feeds p_a to the attention-based Tree-NN to produce the final representation. In this implementation, the sequential RNN and the attention Tree-NN are trained jointly.

3.4 Training

We use the 300-dimensional *Glove vectors* (Pennington et al., 2014) as the initial word representations. Out-of-vocabulary words are randomly initialized for a uniform distribution.

We used AdaGrad (Duchi et al., 2011) for optimization, with a learning rate of 0.05, and a mini-batch size of 25. We also apply the L2 regularization to our model parameters with $\lambda = 10^{-4}$. In our evaluations, the sentiment classifier used is regularized using *dropout* (Hinton et al., 2012) with a dropout rate of 0.5.

Method	r	ρ	MSE
Illinois-LH (Lai and Hockenmaier, 2014)	0.7993	0.7538	0.3692
UNAL-NLP (Jimenez et al., 2014)	0.8070	0.7489	0.3550
Meaning Factory (Bjerva et al., 2014)	0.8268	0.7721	0.3324
ECNU (Zhao et al., 2014)	0.8414	-	-
LSTM (Tai et al., 2015)	0.8528	0.7911	0.2831
Bidirectional LSTM (Tai et al., 2015)	0.8567	0.7966	0.2736
Child-Sum Tree-LSTM (Tai et al., 2015)	0.8676	0.8083	0.2532
combine-skip+COCO (Kiros et al., 2015)	0.8655	0.7995	0.2561
GRU	0.8595	0.7974	0.2689
Bidirectional GRU	0.8662	0.8025	0.2603
Child-Sum Tree-GRU	0.8672	0.8116	0.2573
Static-attention Child-Sum Tree-GRU	**0.8730**	**0.8117**	**0.2426**
Static-attention Child-Sum Tree-LSTM	0.8692	0.8106	0.2528
Dynamic-attention Child-Sum Tree-GRU	0.8684	0.8106	0.2548
Dynamic-attention Child-Sum Tree-LSTM	0.8701	0.8085	0.2524

Table 1: The evaluation metrics of the SICK semantic relatedness subtask are Pearson's r, Spearman's ρ and MSE. The first group is the SemEval 2014 baselines, and the last group is our results.

Method	Fine-grained(%)	Binary(%)
Paragraph-Vec (Le and Mikolov, 2014)	48.7	87.8
CNN-non-static (Kim, 2014)	48.0	87.2
CNN-multichannel (Kim, 2014)	47.4	88.1
DRNN (Irsoy and Cardie, 2014)	49.8	86.6
LSTM (Tai et al., 2015)	46.4	84.9
Bidirectional LSTM (Tai et al., 2015)	49.1	87.5
Child-Sum Tree-LSTM (Tai et al., 2015)	48.4	85.7
N-ary Tree-LSTM (Tai et al., 2015)	**51.0**	88.0
GRU	46.3	86.2
Bidirectional GRU	46.8	85.6
Child-Sum Tree-GRU	47.8	85.1
Static-attention Child-Sum Tree-GRU	48.1	85.5
Static-attention Child-Sum Tree-LSTM	50.3	86.2
Dynamic-attention Child-Sum Tree-GRU	49.1	87.5
Dynamic-attention Child-Sum Tree-LSTM	49.4	**88.3**

Table 2: The test set accuracies on the Stanford Sentiment Treebank. **Fine-grained**: 5-class classification. **Binary**: 2-class classification.

4 Experiments

4.1 Semantic relatedness

First we conduct our experiment on the semantic relatedness SICK dataset (Marelli et al., 2014). This task is to predict a similarity score of a pair of sentences, based on human generated scores. The score values range from 1 to 5, and higher score indicates that the pair of sentences are semantically related. This dataset consists of 9927 sentence pairs with the split of 4500 training pairs, 500 development pairs and 4927 testing pairs. The similarity score $\hat{y}$ is computed by a neural network whose input is a representation pair (h_L, h_R):

$$\begin{aligned} &h_\times = h_L \odot h_R, \quad h_+ = |h_L - h_R|, \\ &h_s = \sigma(W^{(x)} h_\times + W^{(+)} h_+ + b^{(h)}), \\ &\hat{p}_\theta = softmax(W^{(p)} h_s + b^{(p)}), \\ &\hat{y} = r^\intercal \hat{p}_\theta, \end{aligned} \tag{7}$$

where $r^\intercal$ = [1, 2, ..., 5] is an integer vector. We compute target distribution p as a function of prediction scores y given by $p_i = y - \lfloor y \rfloor$ if $i = \lfloor y \rfloor + 1$, $p_i = \lfloor y \rfloor - y + 1$ if $i = \lfloor y \rfloor$, and 0 otherwise. The cost function is the regularized KL-divergence between p and $\hat{p}_\theta$.

The SemEval results are summarized in Table 1. All of our models are able to outperform the best submission in the SemEval 2014 competition. Our attention-based Tree-LSTM performs better than the Child-Sum Tree-LSTM (Tai et al., 2015) thanks to the attention mechanism. Our Static-attention Tree-GRU achieves the state-of-the-art performance in all of the three metrics. All results shows that our attention-based Tree-NNs learn better representations for semantic relatedness.

4.2 Sentiment classification

The Stanford Sentiment Treebank (SST) is a benchmark for sentiment classification. Each sentence from a movie review is parsed as a constituency tree. Each node in a tree is annotated with a sentiment polarity label. This dataset contains two classification tasks: (1) binary classification: positive, negative, and (2) fine-grained classification: very negative, negative, neutral, positive, and very positive. In this experiment, we use the predefined train/dev/test splits of 6920/872/1821 for the binary classification subtask and 8544/1101/2210 for the fine-grained classification subtask. The classification criterion used is the *Negative Log Likelihood.*

The results on sentiment classification task are presented in Table 2. Our attention-based Child-Sum Tree-LSTM obtains the state-of-the-art performance in binary classification. Again, comparison to the corresponding non-attention models, each of our attention-based model achieves better accuracy. In the fine-gained classification task, the Static-attention Tree-LSTM precedes the Child-Sum Tree-LSTM nearly two percents.

5 Conclusion

In this paper, we proposed the attention-based recursive neural networks for better semantic representation. In our experiments, our models outperformed non-attention counterparts and demonstrated the state-of-the-art performance in semantic relatedness and sentiment classification.

References

[Socher et al.2011] Richard Socher, Cliff C. Lin, Andrew Y. Ng and Christopher D. Manning. 2011. Parsing Natural Scenes and Natural Language with Recursive Neural Networks. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*.

[Goller and Kuchler1996] Christoph Goller and Andreas Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. *IEEE International Conference on Neural Networks*. volume 1, pages 347–352.

[Bengio et al.2003] Yoshua Bengio, R. Ducharme, Pascal Vincent and Christian Jauvin. 2003. A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, 3:1137–1155.

[Elman1990] Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive science*. 14(2):179–211.

[Mikolov2012] Tomas Mikolov. 2012. Statistical Language Models Based on Neural Networks. *Ph.D. thesis, Brno University of Technology*.

[Socher et al.2013] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, Andrew Ng and C. Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*

[Tai et al.2015] Kai Sheng Tai, Richard Socher and Christopher D. Manning. 2015. Improved Semantic Representation From tree-based Long Short-Term Memory Networks. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

[Zaremba and Sutskever2014] Wojciech Zaremba and Ilya Sutskever. 2014. Learning to execute.. *arXiv preprint arXiv:1410.4615*.

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

[Chung et al.2014] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv: 1412.3555*.

[Pennington et al.2014] Jeffrey Pennington, Richard Socher and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *EMNLP 2014*.

[Duchi et al.2011] John Duchi, Elad Hazan and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *The Journal of Machine Learning Research* 12:2121–2159.

[Hinton et al.2012] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv: 1207.0580*

[Marelli et al.2014] 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 se- mantic relatedness and textual entailment. In *SemEval 2014*.

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

[Lai and Hockenmaier2014] Alice Lai and Julia Hockenmaier. 2014. Illinois-lh: A denotational and distributional approach to semantics. In *SemEval 2014*.

[Jimenez et al.2014] Jimenez, Sergio, George Duenas, Julia Baquero, Alexander Gelbukh, Av Juan Dios Batiz, and Av Mendizabal. 2014. UNAL-NLP: Combining soft cardinality features for semantic textual similarity, relatedness and entailment. In *SemEval 2014*.

[Bjerva et al.2014] Johannes Bjerva, Johan Bos, Rob van der Goot, and Malvina Nissim. 2014. The meaning factory: Formal semantics for recognizing textual entailment and determining semantic similarity. In *SemEval 2014*.

[Zhao et al.2014] Jiang Zhao, Tian Tian Zhu, and Man Lan. 2014. Ecnu: One stone two birds: Ensemble of heterogenous measures for semantic relatedness and textual entailment. In *SemEval 2014*.

[Kiros et al.2015] Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun and Sanja Fidler. 2015. Skip-Thought Vectors. In *NIPS 2015*.

[Blunsom et al.2014] Phil Blunsom, Edward Grefenstette, Nal Kalchbrenner, et al. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

[Le and Mikolov2014] Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*.

[Kim2014] Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

[Irsoy and Cardie2014] Ozan Irsoy and Claire Cardie. 2014. Deep recursive neural networks for compositionality in language. *Advances in Neural Information Processing Systems (NIPS)*, pages 2096–2104.