Recursive Tree Attention: Improving Semantic Representations with Syntactic Tree Structured Attention Mechanism

Anonymous Author(s)
Affiliation
Address
email

Abstract

Attention mechanism has shown its effectiveness in state-of-the-art methods on various tasks in natural language processing (NLP). However, these methods are still using attention mechanism in plain, linear topological structures while the syntactic structures of natural languages are known to be hierarchical. Modeling in such syntactic structures like syntactic trees is proved to be superior in semantic representation learning. In this paper, to improve semantic representation learning by modelling attention mechanism in syntactic structures, we propose recursive tree attention, a syntactic tree structured attention mechanism which recursively calculates attention weights and summarizes information through constituency parsing trees. The information of the children is cursively summarized to their parent, then the summarized information from the root node is used as the semantic representations of the current sentence. Experimental results on the text classification tasks demonstrate the effectiveness of our proposed attention mechanism. The approaches with recursive tree attention outperform conventional attention mechanism and other syntactic tree based approaches.

1 Introduction

With the development of natural language processing (NLP), attention mechanism (Luong et al., 2015) is proposed and shows its effectiveness on various tasks (Yu et al., 2017; Ott et al., 2018; Ahmed et al., 2019a; Hao et al., 2019; Geng et al., 2020). By giving a query and a set of key-value pairs, the attention mechanism produces a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key. Transformer (Vaswani et al., 2017) is proposed and uses multi-head attention to learn the correlations between the query and keys in different representation subspaces. Bidirectional encoder representations from Transformers (BERT) (Devlin et al., 2018) is further proposed to provide powerful pre-trained semantic representations and can be easily fine-tuned to obtain new state-of-the-art results on many NLP tasks such as question answering and language inference. Bahdanau et al. (2014) propose a use of attention mechanism with the encoder-decoder framework (Cho et al., 2014) in machine translation which dynamically summarizes the most related features from the encoder outputs with the current decoding state.

However, these attention mechanisms directly compute the attention weights between the query and all available keys in plain topological structures, while the underlying construction process of natural languages is known to be hierarchical (Frege, 1892; Nguyen et al., 2020). Sentence-level syntactic

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

parsing uncovers these hierarchical structural relations and discloses the rules that words are used to form sentences (Zhang, 2020), which can be beneficial for a number of NLP tasks (Yamada and Knight, 2001; Chan and Roth, 2011; Zou et al., 2015). Modeling in such syntactic structures like syntactic trees is proved to be superior in semantic representation learning (Tai et al., 2015; Roy et al., 2020). By noticing the superiority of attention mechanism on summarizing information and these internal structural relations provided by syntactic parsing, it is of possibility to improve semantic representation learning with syntactic tree structured attention mechanism.

However, researches on syntactic tree structured attention mechanism is still in its infancy. Recent works processing syntactic tree structured data with attention mechanism are mainly introducing additional modules or inputs to provide syntactic structural information, rather than directly modeling attention mechanism in syntactic structures. Wang et al. (2019) added an additional constituent attention module to encourage the attention heads to follow tree structures. Harer et al. (2019) inserted a parent-sibling tree convolution block to provide syntactic information. Nguyen et al. (2020) proposed a work to use the hierarchical accumulation of word embeddings of the non-terminal nodes in syntactic trees as additional inputs for Transformer.

In this paper, to improve semantic representation learning by syntactic tree structured modeling rather than only providing syntactic information to attention mechanism, we propose recursive tree attention, a syntactic tree structured attention mechanism which recursively summarizes the information through constituency parsing trees. For each node in the syntactic tree, the information of its children are summarized to this node with attention weights calculated by an inner plain structured attention mechanism. Positional and constituent information are also provided by adding and concatenating positional and constituent embeddings when calculating attention weights. Then the summarized information from the root node is used as the semantic representation of the current sentence.

The proposed recursive tree attention is compatible and can be used as an in-place replacement for the original attention mechanism in many occasions. Experimental results on the text classification tasks demonstrate the effectiveness of our proposed recursive tree attention. The approaches with recursive tree attention outperform the plain structured attention mechanism based and other state-of-the-art syntactic tree based approaches.

2 Related works

2.1 Attention mechanism

Attention mechanism (Luong et al., 2015) has been widely researched in the NLP area. The attention mechanism calculates the attention weights between a query and a set of keys, then outputs a weighted sum of the values, where the weight for each value is obtained from the corresponding keys.

Many variants of attention mechanisms are proposed with different kinds of improvements. Scaled dot-product attention (Vaswani et al., 2017) is proposed to scale the attention weights to avoid extremely small gradients. Additive attention (Bahdanau et al., 2014) is proposed to compute the compatibility function using a feed-forward network. Location sensitive attention (Chorowski et al., 2015) extends the additive attention mechanism to use cumulative attention weights as additional feature in the encoder-decoder framework.

However, these attention mechanisms directly compute the attention weights between the query and all available keys in plain topological structures, missing the hierarchical structural information in natural languages.

2.2 Hierarchical attention mechanism

Hierarchical structured attention mechanism has outperformed plain structured attention mechanism in many researches (Yang et al., 2016; Li et al., 2018). Basic hierarchical structured attention is proposed for document classification where attention mechanisms are applied at word and sentence levels to attend different content when constructing document representations (Yang et al., 2016). Feature

pyramid attention (Li et al., 2018) is proposed for image semantic segmentation by constructing a pyramid-like structure to improve the learnt feature representation.

However, these works are using primary structures rather than syntactic structures of natural languages. The syntactic structures of natural languages are more complicated than the hierarchical structures used in these works.

2.3 Tree-LSTM and attentive Tree-LSTM

Tree-LSTM (Tai et al., 2015) is proposed to model long-short term memory (LSTM) (Hochreiter and Schmidhuber, 1997) network in syntactic tree structure to improve semantic representation learning. Attentive Tree-LSTM (Zhou et al., 2016) is further proposed and integrates a generalized attention framework inside Tree-LSTM cells, using attention mechanism to generate the hidden state for the current step from the hidden states of its children.

However, these works are still using the original input sequence as the inputs for the each step rather than the context vectors dynamically summarized by attention mechanism. Bahdanau et al. (2014) have proved that using dynamically summarized context vectors is superior than directly using the original input sequence for recurrent neural networks (RNNs) (Rumelhart et al., 1986). Moreover, constituency information which can further improve semantic representation learning is missing in these works.

2.4 Tree attention and tree Transformer

Many works attempt to improve attention mechanism to process syntactic tree structured data. The names of these works are all similar to "tree attention" or "tree Transformer". However, the attention mechanisms used in these works are still plain structured. To our best knowledge, no syntactic tree structured attention mechanism is proposed in these works.

The tree attention proposed by Ahmed et al. (2019b) is actually a more generalized form of the attention mechanism in the attentive Tree-LSTM to make the model compatible with both the constituency and dependency parsing trees.

The latent tree attention proposed by Bradbury and Socher (2017) produces the context vector between the stacks of the encoder and decoder StackLSTM (Dyer et al., 2015) networks. However, the attention mechanism is still plain structured, which simultaneously computes attention weights for all nodes in syntactic trees.

The Tree Transformer proposed by Wang et al. (2019) inserts an additional constituent attention module to the encoder of Transformer to encourage the attention heads to follow tree structures. However, neither syntactic information nor syntactic structured modeling is used in the decoder of this work.

The Tree-Transformer proposed by Harer et al. (2019) inserts a parent-sibling tree convolution block to the encoder and decoder of Transformer to provide the syntactic information. However, the provided syntactic information is limited to the parent and the left sibling. The information from the children and other siblings are not considered.

The TreeTransformer proposed by Nguyen et al. (2020) calculates hierarchical accumulations as the representations for non-terminal nodes in the syntactic trees from the representations of all its successors. The representations of nodes are divided into two groups the leaf and non-terminal nodes. Then the representations of these two groups are used as two separate inputs to the encoder and decoder of Transformer. Both the self relations inside and cross relations between the two groups are considered when calculating attention weights. However, without structured modelling, only dividing nodes into two groups may not take full use of the structural information in syntactic trees.

3 Methodology

In this section, we will describe the details of our proposed recursive tree attention mechanism in Section 3.1. Then we introduce a use of recursive tree attention for text classification in Section 3.2.

3.1 Recursive tree attention

3.1.1 Model description

Similar to the original attention mechanism, the proposed recursive tree attention mechanism also computes the correlations between query and key vectors and then produces a weighted sum of the corresponding value vectors, and can be formulated as:

$$C = A_{tree}(Q, K, V) \tag{1}$$

where A_{tree} is the proposed recursive tree attention mechanism, Q, K and V are the query, keys and values, and C is the weighted sum node which is also known as the context vector. Particularly, K can have the keys for both the leaf and non-terminal nodes or just the leaf nodes, and V should only have the values for the leaf nodes.

Instead of directly calculating the attention weights between the given query and all available keys, the proposed tree attention recursively calculates the attention weights through the constituency parsing tree and summarizes the information from the leaf nodes to the root node. Then the context vector of the root node is used as the semantic information of the current sentence. An illustration of how the recursive tree attention works on a positive-negative text classification task is shown in Figure 1.

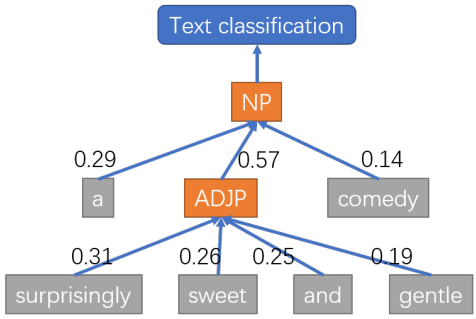


Figure 1: An illustration of recursive tree attention on a positive-negative text classification task. The information of the leaf nodes are recursively summarized to the root node. Some information such as constituent labels of leaf nodes is omitted in this figure. The shown attention weights are from a real evaluation.

The context vector of the root node can be calculated by the following recursive algorithm. For each non-terminal node, attention weights are calculated between the given query and the keys of its direct children by an inner plain structured attention mechanism such as dot product attention (Luong et al., 2015) or additive attention (Bahdanau et al., 2014). Then the weighted sum of the context vectors of its children is used as the context vector for this node.

Positional and constituent information are also included by adding and concatenating positional and constituent embeddings to query and keys of the inner attention mechanism respectively. Positional encoding is added to the keys, which we will describe in Section 3.1.2. The constituent labels of each nodes are represented as constituent embeddings by a 64 dimensional embedding layer. Then the constituent embeddings of the current and children nodes are respectively concatenated to the query and the keys of children to represent the constituent and syntactic information.The above procedure can be formulated as:

$$C_{node} = A_{plain}([Q; L_{node}], [K_{children} + P; L_{children}], C_{children}) \tag{2}$$

where A_{plain} is the inner plain structured attention mechanism, L_{node} and $L_{children}$ are the constituent embeddings of the node and its children, $K_{children}$ and $C_{children}$ are the keys and context vectors of its children, P is the positional encoding and C_{node} is the output context vector for this non-terminal node.

The keys for non-terminal nodes in the parsing tree can be given in K as inputs or or recursively generated from the keys of leaf nodes by the following equation:

$$K_{node} = A_{plain}([Q; L_{node}], [K_{children} + P; L_{children}], K_{children}) \tag{3}$$

where K_{node} is the key for this non-terminal node. Particularly, we can directly use the attention weights calculated in Equation 2 to get K_{node} since the query and keys for the inner attention are the same.

However, if the keys for non-terminal nodes are given in K, the attention weights for all nodes in the parsing tree can be calculated in parallel. This allow that the proposed recursive tree attention can be employed in more frameworks like the encoder-decoder framework without bringing significant time for computing.

Finally for each leaf node in the parsing tree, its key and value are given in K and V. Its context vector is equal to its value. This can be shown as:

$$C_{leaf} = V_{leaf} \tag{4}$$

where C_{leaf} and V_{leaf} are the context vector and value of the leaf node.

A simplified form of Equation 2 and 3 can be obtained when only the keys and values for the leaf nodes are given and they are same. In this case, K_{node} and C_{node} are equal for all nodes in the parsing tree. And Equation 2 and 3 become:

$$C_{node} = A_{plain}([Q; L_{node}], [C_{children} + P; L_{children}], C_{children}) \tag{5}$$

which can be directly used to summarize of the values of the words to sentence-level semantic representations for many NLP tasks like text classification.

3.1.2 Positional encoding

The positions of nodes in parsing trees are carefully designed. The position for each node in the parsing tree is obtained by the following algorithm. Firstly we temporally set the positions of all leaf nodes as the positions of their corresponding words. Then for each non-terminal node, we temporally set its position to the minimum of the positions of all its successors. Thereafter, the temporal position of each node is subtracted by the minimum positions of the current node and its siblings. Particularly, if a node has no sibling, its position will minus itself and become zero. An illustration of how the position for each node is calculated is depicted in Figure 2.

(a) (b)

Figure 2: An illustration of the position calculation algorithm. (a) Temporally set the position of each node to the minimum of all its successors. (b) The temporal position of each node is subtracted by the minimum positions of the current node and its siblings.

The position for each node is the relative beginning in the word sequence of its parent. Therefore, there is a conclusion that for any leaf node in the sequence, if we add the positions through its path to

the root, the sum will equal to the position of the corresponding word. In other words, the positional information of each words is split into their corresponding nodes in parsing trees.

We then employ the positional encoding in Transformer (Vaswani et al., 2017) to encode the calculated positions, which is computed as:

$$P_{(pos,2i)} = sin(pos/10000^{2i/d}) \tag{6}$$

$$P_{(pos,2i+1)} = cos(pos/10000^{2i/d}) \tag{7}$$

where pos is the position, i is the dimension, and d is the number of the feature dimensions.

3.1.3 Extended parsing tree for sub-words

Recently algorithms like WordPiece (Wu et al., 2016) are more often to be used in NLP (Devlin et al., 2018; Tan and Bansal, 2019; Bao et al., 2020) since dividing words into a limited set of common sub-word units can greatly reduce errors than phrase-based systems in tasks like machine translation.

However, extra changes are required to use sub-words with the proposed recursive tree attention since parsing trees can only be obtained at word level. We add extra nodes and constituent labels for the divided sub-words.The nodes of sub-words are connected to their corresponding words in the parsing tree. And new constituent labels for sub-words are created based on the constituent labels of their corresponding words. In other words, we add extra grammars for all possible terminal constituent labels and all possible lengths to the original grammar set like:

$$N \rightarrow N_{SW} \dots N_{SW} \tag{8}$$

$$V \rightarrow V_{SW} \dots V_{SW} \tag{9}$$

$$ADJ \rightarrow ADJ_{SW} \dots ADJ_{SW} \tag{10}$$

$$\dots$$

where N, V, ADJ are terminal constituent labels and N_{SW}, V_{SW}, ADJ_{SW} are new constituent labels for their corresponding sub-words.

3.2 Text classification with recursive tree attention

In this section, we introduce a use of recursive tree attention for text classification, which is shown in Figure 3 .

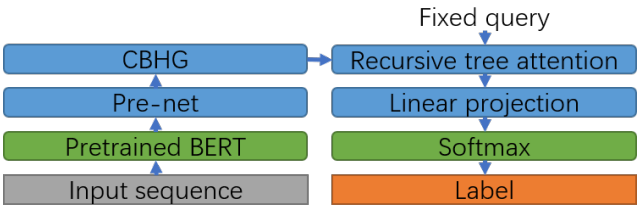


Figure 3: Architecture of the proposed use of recursive tree attention for text classification.

Word embeddings extracted by a pre-trained BERT model at sub-word level are used as inputs. Therefore the extra constituent labels and nodes discribed in Section 3.1.3 are applied. Then the word embeddings are passed through a pre-net and a CBHG module (Wang et al., 2017). The pre-net consists of two fully connected layers which have 128 and 256 hidden units respectively with dropout probability 0.5. The CBHG module consists of a bank of 1-D convolutional filters, a max pooling layer, four highway networks (Srivastava et al., 2015) and a 256 dimensional bidirectional GRU (Cho et al., 2014) network.

Then recursive tree attention is used to summerize the representations extracted by the CBHG module to a sentence-level context vector. The recursive tree attention is developed with Equation 5, in which

the keys and values are always equal. The query is set to a fixed vector to find the most related features to the current classification task, like a random fixed vector a zero vector for additive attention.

The summarized context vector from the root node is used as the semantic representation of the current sentence. Then it is linear projected to a vector followed by the softmax operation to predict the class.

4 Experiments

We evaluate our proposed recursive tree attention in the text classification tasks with the approach described in Section 3.2. Training setups and experimental results are posted in Section 4.1 and Section 4.2. Ablation studies are also conducted in Section 4.3.

4.1 Training setup

We implement the model discribed in Section 3.2 as our proposed approach for text classification. We use an 128 dimensional additive attention (Bahdanau et al., 2014) as the inner attention used in recursive tree attention. We use a zero vector as the fixed query for recursive tree attention. The input BERT representations are 768 dimensional vectors extracted by a pre-tained BERT model (Wolf et al., 2020). The parameters of the pre-net and the CBHG modules are same to those employed in Tacotron (Wang et al., 2017).

We train and evaluate our model with the Stanford Sentiment Treebank (SST) (Socher et al., 2013) dataset on the binary classification and the fine-grained classification over five classes tasks (Tai et al., 2015). We also follow the data separation for training, validation and testing by Tai et al. (2015) to enable a fair comparison with other approaches.

We also train and evaluate our model with the IMDB movie review dataset (Maas et al., 2011). An addtional plain structured additive attention layer is inserted before the fully connected layer to summarize the context vector of the multiple sentences in a paragraph. We use the Berkeley neural parser (Kitaev et al., 2019; Kitaev and Klein, 2018) to obtain the constituency parsing trees for the IMDB corpus.

The model is optimized with cross entropy as the loss function. The learning rate is fixed to 10^{-5}. The model is developed with PyTorch (Paszke et al., 2019). The model has 7.0M parameters and trained for 10,000 steps for about 10 hours on a NVIDIA 3080 GPU with a batch size of 256.

4.2 Experimental results

The results are shown in Table 1 with accuracy as the metric. We compare our model with an bidirectional LSTM network based approach (Nguyen et al., 2020), Tree-LSTM (Tai et al., 2015), Transformer (Vaswani et al., 2017) and the TreeTransformer by Nguyen et al. (2020). All the syntactic tree based models outperform sequential models, showing the effectiveness of syntactic information. The proposed model also outperforms all the other models, including other state-of-the-art syntactic tree based models on all tasks.

4.3 Ablation studies

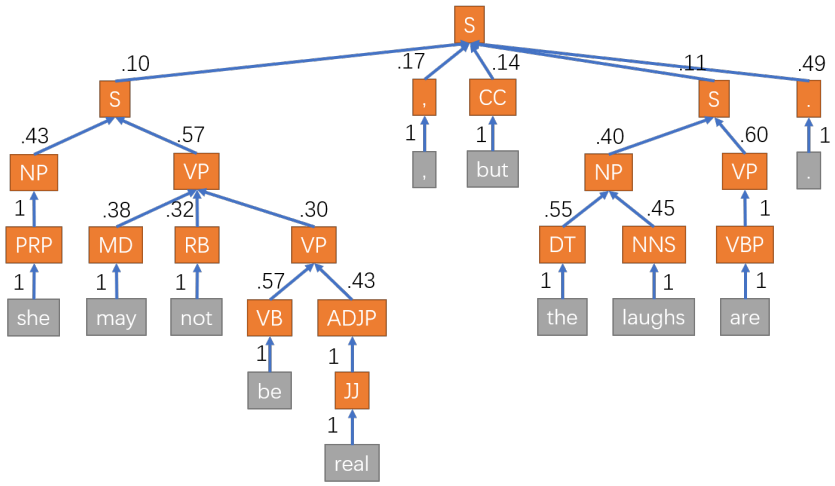
4.3.1 Effectiveness of structured modeling in recursive tree attention

We explore the effectiveness of structured modeling in recursive tree attention by replacing the attention mechanisms in the proposed models with plain structured attention mechanism. Evaluations of this variant are conducted and shown in the sixth line of Table 1. Without structured modeling, the performances greatly drop on both the tasks and are even worse than the sequential models on some tasks. This result clearly shows that the performances of the proposed approaches are mainly benefited by the syntactic tree structured attention mechanism rather than the inputs extracted by pre-trained BERT.

Table 1: Accuracy on SST-2 and SST-5 text classification tasks by different approaches.

Approaches	**SST-2**	**SST-5**	**IMDB**
BLSTM	76.0	35.1	85.8
Transformer	74.8	37.6	86.5
Tree-LSTM	82.0	43.9	-
TreeTransformer (Nguyen et al., 2020)	84.3	47.4	90.1
Proposed	**89.1**	**48.7**	**91.2**
- without structured modeling	80.2	36.7	-
- without positional encoding	87.5	47.7	-
- without constituent embedding	87.1	46.0	-
- without extra sub-word nodes and constituent labels	85.2	36.5	-

A comparison between the attention weights learnt by the proposed recursive tree attention mechanism and plain structured attention mechanism is also shown in Figure 4. Comparing to the plain structured attention mechanism, the proposed recursive tree attention mechanism assigns much more attention weights to the constituents like conjunctions and punctuations. Meanwhile, the attention weights for the conjunctions and punctuations are below average in plain structured attention. This result shows that the proposed recursive tree attention are more good at capturing syntactic information to improve semantic representations.



(a) Attention weights learnt by the proposed recursive tree attention mechanism.

(b) Equivalent attention weights learnt by the proposed recursive tree attention for each words.

(c) Attention weights learnt by plain structured attention mechanism.

Figure 4: The attention weights learnt by the proposed recursive tree attention mechanism and plain structured attention mechanism for sentence "she may not be real, but the laughs are ." in the SST-2 task. (a) The attention weights learnt by the proposed recursive tree attention mechanism. The extra sub-word nodes are omitted since the sub-words are same to the words for this sentence. (b) Equivalent attention weights learnt by the proposed recursive tree attention for each words. The equivalent attention weight for each word is gained by multiplying the attention weights through its path to the root. (b) The attention weights learnt by plain structured attention mechanism.

We also notice that the full stop at the end of the sentence has a much greater attention weight than the others. There are two reasons that may explain it. First reason is that the last punctuation does

be important to classify this sentence. The second one is that the positional encoding of the last full stop constituent node are also providing length information to the previous sentence constituent. Therefore it may not only have the attention weight of itself, but also part of the attention weight of the previous constituent.

4.3.2 Effectiveness of the positional information in recursive tree attention

We explore the effectiveness of the positional information in recursive tree attention by excluding the positional encoding when calculating attention weights. Evaluations of this variant are conducted and shown in the seventh line of Table 1. The performances slightly drop when compared with the proposed model. This result demonstrate the the effectiveness of the positional information.

4.3.3 Effectiveness of the constituent information in recursive tree attention

We explore the effectiveness of the constituent information in recursive tree attention by excluding the constituent embeddings when calculating attention weights. Evaluations of this variant are conducted and shown in the eighth line of Table 1. The performances also drop when compared with the proposed model and are worse than the variants which excludes the positional information. And it can be also concluded that constituent information is more important than positional information for recursive tree attention.

4.3.4 Effectiveness of extra nodes and constituent labels for sub-words

We explore the effectiveness of the extra nodes and constituency labels for sub-words described in Section 3.1.3 by directly using the average of the embeddings of sub-words as the embedding of the corresponding word. Evaluations of this variant are conducted and shown in the last line of Table 1. The performances greatly decrease after using the averages as word embeddings. This result shows that the sub-words are holding different parts of information of words. The proposed recursive tree attention can attend to the required parts by using extra nodes and constituent labels and greatly improve the performance.

5 Conclusions and future directions

5.1 Conclusions

In this paper, we propose recursive tree attention, a syntactic tree structured attention mechanism which recursively calculates attention weights and summarizes information through constituency parsing trees. Not like the other works which use additional modules or inputs to present the syntactic structure information, our work uses structured modeling to change the procedure of calculating attention weights.

Experimental results on the text classification tasks demonstrate the effectiveness of our proposed recursive tree attention. The approaches with recursive tree attention outperform conventional attention mechanism and several syntactic tree based approaches.

5.2 Future directions

The proposed recursive tree attention is compatible and can be used as an in-place replacement of the original attention mechanism in many occasions. We are planning to apply recursive tree attention to more tasks such as machine translation and even to non NLP tasks such as speech synthesis.

We notice that recursive tree attention may also suit the uses of attention mechanism in Transformer. However, there are still some issues which need to be solved such as the computational complexity. We will try to solve these issues and apply recursive tree attention in Transformer.

For now recursive tree attention needs to explicitly input the syntactic trees. Integrating syntactic parsing into recursive tree attention is also a future directions of our researches.

References

M. Ahmed, J. Islam, M. R. Samee, and R. E. Mercer. 2019a. Identifying Protein-Protein Interaction Using Tree LSTM and Structured Attention. In *2019 IEEE 13th International Conference on Semantic Computing (ICSC)*, pages 224–231. ISSN: 2325-6516.

M. Ahmed, M. R. Samee, and R. E. Mercer. 2019b. Improving Tree-LSTM with Tree Attention. In *2019 IEEE 13th International Conference on Semantic Computing (ICSC)*, pages 247–254. ISSN: 2325-6516.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv:1409.0473 [cs, stat]*. ArXiv: 1409.0473.

Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Jianfeng Gao, Songhao Piao, Ming Zhou, and Hsiao-Wuen Hon. 2020. UniLMv2: Pseudo-Masked Language Models for Unified Language Model Pre-Training. In *Proceedings of the 37th International Conference on Machine Learning*, pages 642–652. PMLR. ISSN: 2640-3498.

James Bradbury and Richard Socher. 2017. Towards Neural Machine Translation with Latent Tree Attention. *arXiv:1709.01915 [cs]*. ArXiv: 1709.01915.

Yee Seng Chan and Dan Roth. 2011. Exploiting syntactico-semantic structures for relation extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 551–560, USA. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv:1406.1078 [cs, stat]*. ArXiv: 1406.1078.

Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 577–585, Cambridge, MA, USA. MIT Press.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]*. ArXiv: 1810.04805.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-Based Dependency Parsing with Stack Long Short-Term Memory. 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 334–343.

Gottlob Frege. 1892. Über sinn und bedeutung. *Zeitschrift für Philosophie und philosophische Kritik*, 100:25–50.

ZhiQiang Geng, GuoFei Chen, YongMing Han, Gang Lu, and Fang Li. 2020. Semantic relation extraction using sequential and tree-structured LSTM with attention. *Information Sciences*, 509:183–192.

Jie Hao, Xing Wang, Shuming Shi, Jinfeng Zhang, and Zhaopeng Tu. 2019. Multi-Granularity Self-Attention for Neural Machine Translation. *arXiv:1909.02222 [cs]*. ArXiv: 1909.02222.

Jacob Harer, Chris Reale, and Peter Chin. 2019. Tree-Transformer: A Transformer-Based Method for Correction of Tree-Structured Data. *arXiv:1908.00449 [cs, stat]*. ArXiv: 1908.00449.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Nikita Kitaev, Steven Cao, and Dan Klein. 2019. Multilingual Constituency Parsing with Self-Attention and Pre-Training. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3499–3505, Florence, Italy. Association for Computational Linguistics.

Nikita Kitaev and Dan Klein. 2018. Constituency Parsing with a Self-Attentive Encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia. Association for Computational Linguistics.

Hanchao Li, Pengfei Xiong, Jie An, and Lingxue Wang. 2018. Pyramid Attention Network for Semantic Segmentation. *arXiv:1805.10180 [cs]*. ArXiv: 1805.10180.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. *arXiv:1508.04025 [cs]*. ArXiv: 1508.04025.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning Word Vectors for Sentiment Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Xuan-Phi Nguyen, Shafiq Joty, Steven C. H. Hoi, and Richard Socher. 2020. Tree-structured Attention with Hierarchical Accumulation. *arXiv:2002.08046 [cs]*. ArXiv: 2002.08046.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling Neural Machine Translation. *arXiv:1806.00187 [cs]*. ArXiv: 1806.00187.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Deboleena Roy, Priyadarshini Panda, and Kaushik Roy. 2020. Tree-CNN: A hierarchical Deep Convolutional Neural Network for incremental learning. *Neural Networks*, 121:148–160.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536. Number: 6088 Publisher: Nature Publishing Group.

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, Seattle, Washington, USA. Association for Computational Linguistics.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway Networks. *arXiv:1505.00387 [cs]*. 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, Beijing, China. Association for Computational Linguistics.

Hao Tan and Mohit Bansal. 2019. LXMERT: Learning Cross-Modality Encoder Representations from Transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5100–5111, Hong Kong, China. Association for Computational Linguistics.

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 I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Yaushian Wang, Hung-Yi Lee, and Yun-Nung Chen. 2019. Tree Transformer: Integrating Tree Structures into Self-Attention. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1061–1070, Hong Kong, China. Association for Computational Linguistics.

Yuxuan Wang, R. J. Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, Quoc Le, Yannis Agiomyrgiannakis, Rob Clark, and Rif A. Saurous. 2017. Tacotron: Towards End-to-End Speech Synthesis. *arXiv:1703.10135 [cs]*. ArXiv: 1703.10135.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv:1609.08144 [cs]*. ArXiv: 1609.08144.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, ACL '01, pages 523–530, USA. Association for Computational Linguistics.

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.

Licheng Yu, Mohit Bansal, and Tamara Berg. 2017. Hierarchically-Attentive RNN for Album Summarization and Storytelling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 966–971.

MeiShan Zhang. 2020. A survey of syntactic-semantic parsing based on constituent and dependency structures. *Science China Technological Sciences*, 63(10):1898–1920.

Yao Zhou, Cong Liu, and Yan Pan. 2016. Modelling Sentence Pairs with Tree-structured Attentive Encoder. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2912–2922, Osaka, Japan. The COLING 2016 Organizing Committee.

Huang Zou, Xinhua Tang, Bin Xie, and Bing Liu. 2015. Sentiment Classification Using Machine Learning Techniques with Syntax Features. In *2015 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 175–179.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]