# Tree-LSTM Guided Attention Pooling of DCNN for Semantic Sentence Modeling

Liu Chen[1,3](✉), Guangping Zeng[1,3], Qingchuan Zhang[2,3], and Xingyu Chen[1,3]

[1] School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing, China
chenliueve@163.com, zgp@ustb.edu.cn, cscserer@sina.com

[2] School of Computer and Information Engineering, Beijing Technology and Business University, Beijing, China
zqc1982@126.com

[3] Beijing Key Laboratory of Knowledge Engineering for Materials Science, Beijing, China

**Abstract.** The ability to explicitly represent sentences is central to natural language processing. Convolutional neural network (CNN), recurrent neural network and recursive neural networks are mainstream architectures. We introduce a novel structure to combine the strength of them for semantic modelling of sentences. Sentence representations are generated by Dynamic CNN (DCNN, a variant of CNN). At pooling stage, attention pooling is adopted to capture most significant information with the guide of Tree-LSTM (a variant of Recurrent NN) sentence representations. Comprehensive information is extracted by the pooling scheme and the combination of the convolutional layer and the tree long-short term memory. We evaluate the model on sentiment classification task. Experiment results show that utilization of the given structures and combination of Tree-LSTM and DCNN outperforms both Tree-LSTM and DCNN and achieves outstanding performance.

**Keywords:** Dynamic Convolutional Neural Network (CNN) · Tree-Structured Long-Short Term Memory (Tree-LSTM) · Attention Pooling · Semantic Sentence Modeling

## 1 Introduction

The sentence modelling problem is an essential component of natural language processing (NLP) and has drawn mass attention recently. The objective of sentence modelling is to analyze and represent the semantic content of a sentence for purposes of sentiment analysis, document summarization, machine translation, discourse analysis, etc. [1]. Sentence features, the key of sentence modeling, are usually extracted from features of word representations.

With advances in word vector representation [2, 3], word vectors become a common practice of word representation for classification. Vector representations of words can even preserve the semantic relationship [4]. In the vector space, words with similar semantics lie close in Euclidean or cosine distance.

Recently, neural network methods have achieved outstanding performance namely recursive neural networks (Recursive NN) [5], recurrent neural networks (Recurrent NN) [6] and convolutional neural networks (CNN) [7].

The recursive neural networks of [5, 8, 9] rely on syntactic parse trees and recursively compose sentence representation from child nodes in the parse tree. The learning performance of recursive neural networks depends much on the construction of the textual tree which can be quite time-consuming.

The recurrent neural networks of [10, 11] compose word vectors from one end to the other stores the information of all previous contexts in a fixed-sized hidden layer. RNNs with Long Short-Term Memory (LSTM) units [12] have re-emerged as a popular architecture due to their representational power and effectiveness at dealing with vanishing gradient problem and capturing long-term dependencies. Studies of [12, 13] proposed an improved variant of Recurrent NN which integrate Recursive NNs by feeding syntactic parse tree into LSTM.

CNN was originally proposed in computer vision [14], and recently it becomes popular in NLP tasks. Different from Recursive and Recurrent NNs, CNN encodes word vectors by convolution operation and generates a fixed-sized high-level representation by pooling. [7] proposed a Dynamic Convolutional Neural Network (DCNN) with multiple layers of convolutional and dynamic pooling operations which handles input sentences of varying length and forms feature maps which is capable of explicitly capturing syntactic or semantic relations between nonconsecutive parts within the input sentence.

In this paper, we introduce an attention pooling dynamic CNN architecture (abbreviated to AP-DCNN) to combine the benefits of CNN, Recursive and Recurrent NNs with attention pooling schema. The AP-DCNN can be considered a variant of APCNN in [15] where the attention pooling scheme is proposed. The Tree-LSTM model [16] is employed to enhance the information extraction capability of the pooling layer. The Tree-LSTM model is also concatenated with the convolutional structure to extract comprehensive information, namely historical, future and local context information, of any position in a sequence at the testing phase.

We conduct experiments on sentiment analysis task of Stanford Treebank Datasets and results show that utilization of the given structures and combination of Tree-LSTM and DCNN leads to better performance.

## 2 The Proposed AP-DCNN Model

The architecture of the AP-DCNN model is shown in Fig. 1. The following subsections describe the proposed model in detail.

### 2.1 Word Embedding

The input of the model is $N$ sentences with variable lengths. Each sentence $S$ is constituted by words which are represented by vectors. Recent researches have demonstrated that continuous word representation is a popular and powerful method for sentence classification tasks. A word vector can be formed as follows:

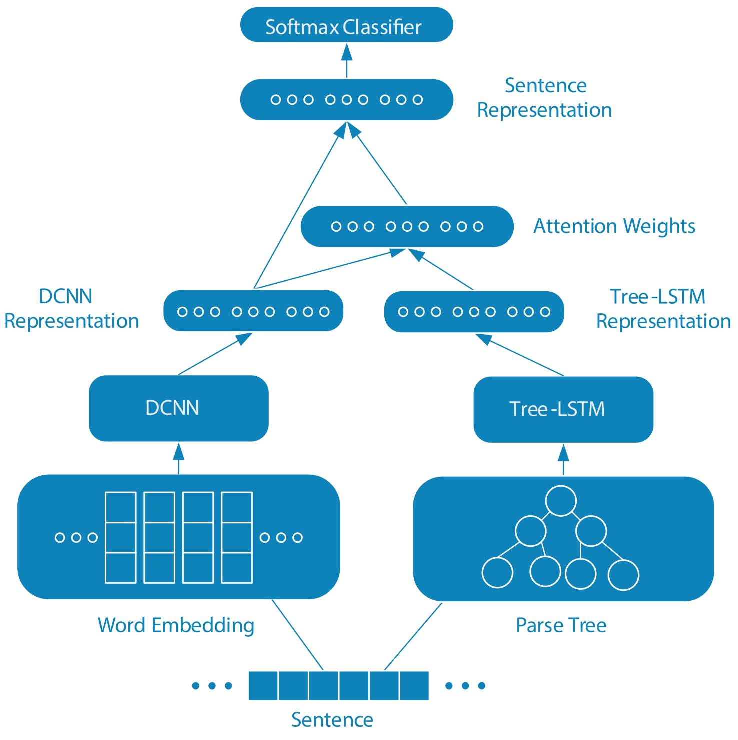

**Fig. 1.** Architecture of AP-DCNN. Convolution filters perform convolutions on the input sentence matrix to generate local Sentence representations. An attention pooling scheme is used to integrate local representations into the final sentence representation with attention weights. These weights are composed by comparing local representations with Tree-LSTM based sentence representation by position and optimized during the training phase.

$$x = Wp, \tag{1}$$

where $p \in \mathbb{R}^V$ is a one-hot vector where the position the word appears is 1 while the others are 0, $W \in \mathbb{R}^{d \times V}$ is a word-representation matrix, in which the $i$ th column is the vector representation of the $i$ th word in the vocabulary, and V denotes the vocabulary size.

We adopt the publicly available word2vec vectors as initial word embedding matrices to make adequately use of semantic and grammatical associations of words. The vectors, with dimension of 300, are trained on 100 billion words from the Google News by using the continuous bag-of-words method [3] and maximizing the average log probability of all the words [17]. Words not present in the set of pre-trained words are initialized randomly.

## 2.2 DCNN Based Sentence Representation

Convolutional layers play significant roles in the success of the CNN because they can encode significant information contained in input data with significantly fewer parameters than other deep learning architectures. We adopt the Dynamic Convolutional Neural Network (DCNN) for the semantic modelling of sentences. The network uses Dynamic *k*-Max Pooling [7], a global pooling operation over linear sequences. The network handles input sentences of varying length and induces a feature graph over the sentence which is capable of explicitly capturing short and long-range relations.

The convolution operation at layer $l$ is conducted between $k_l$ filters $W_l^T \in \mathbb{R}^{md \times k_l}$ and a concatenation vector $c_{l-1_{i:i+m-1}}$ which represents a window of m features starting from the $i^{\text{th}}$ feature in the feature maps conducted in the last layer while $l = 1$ represents the word embedding in the origin sentence.

The term $d$ is the dimension of word embedding. Multiple filters with differently initialized weights are used to improve the model's learning capability.

The number of filters $k_{top}$ is determined using cross-validation and the convolution operation is governed by:

$$c_{l_i} = g\left(W_l^T c_{l-1_{i:i+m-1}} + b_l\right) \in \mathbb{R}^{k_l}, \tag{2}$$

where $k_l$ denotes the number of filters in the current convolutional layer $l$, using:

$$k_l = \max\left(k_{top}, \left\lceil \frac{L-l}{L} T \right\rceil\right), \tag{3}$$

where $L$ is the total number of convolutional layers in the network.

Following the approach in [4, 18], we also use filters with varying convolution window sizes to form parallel DCNNs so that they can learn multiple types of embedding of local regions so as to complement each other to improve model accuracy. Sentence representations produced by all the distinct DCNNs are concatenated to form the final feature vector as an input to the top SoftMax classifier.

## 2.3 Tree-LSTM Based Sentence Representation

Tree-LSTM is a variant of LSTM. A Tree-LSTM unit (indexed by $j$) contains input and output gates $i_j$ and $o_j$, a memory cell $c_j$ and hidden state $h_j$ as a standard LSTM do, while contains one forget gate $f_{jk}$ for each child $k$, which is the difference. This allows the Tree-LSTM unit to selectively incorporate information from each child. A Tree-LSTM unit at each node takes an input word vector $\mathrm{x}_j$ which depends on the tree structure used for the network. Given a tree, let $C(j)^K$ denote the set of $K$ children of node $j$. The transition proceeds as follows:

$$i_j = \sigma\left(W^{(i)}x_j + \sum\nolimits_{k \in C(j)^K} U_k^{(i)} h_{jk} + b^{(i)}\right), \tag{4}$$

$$f_{jk} = \sigma\left(W^{(f)}x_j + \sum\nolimits_{l \in C(j)^K} U_{kl}^{(i)} h_{jl} + b^{(f)}\right), \tag{5}$$

$$o_j = \sigma\left(W^{(o)}x_j + \sum\nolimits_{k \in C(j)^K} U_k^{(o)} h_{jk} + b^{(o)}\right), \tag{6}$$

$$u_j = tanh\left(W^{(u)}x_j + \sum\nolimits_{k \in C(j)^K} U_k^{(u)} h_{jk} + b^{(u)}\right), \tag{7}$$

$$c_j = i_j \odot u_j + \sum\nolimits_{k \in C(j)^K} f_{jk} \odot c_k, \tag{8}$$

$$h_j = o_j \odot tanh\left(c_j\right), \tag{9}$$

where in Eq. (5), $k \in C(j)^K$.

We use binary tree LSTM ($K = 2$) because it suits more to constituency trees we use. When the forgetting node has only one child, the model can be considered as the standard LSTM. We denote the sentence representation as $\tilde{s}$.

## 2.4 Attention Pooling

In the attention pooling stage, we compare DCNN-based sentence representation and Tree-LSTM based sentence representation to calculate the attention weights. By controlling the output dimension of the Tree-LSTM same as the number of convolutional filters $k_{top}$, we are able to map the both representations into the space of the same dimension. The higher the similarity between the DCNN sentence representation and Tree-LSTM representation, the bigger attention weight is assigned to DCNN representation. The attention weights is given by:

$$\alpha_i = \frac{exp(sim(c_i, \tilde{s}))}{\sum_{i=1}^{T} exp(sim(c_i, \tilde{s}))}. \tag{10}$$

The function sim denotes a method to measure similarity between inputs where cosine similarity is adopted. The final sentence representation guided by the attention weights is calculated by:

$$s = \alpha \odot c \in \mathbb{R}^T. \tag{11}$$

The final sentence representation forms the input of the top classifier.

## 2.5 Softmax Classifier

The sentence representation s is naturally regarded as an input to the top classifier during the training phase while $[\mathrm{s}, \tilde{s}]$ is used at the testing phase. A linear transformation layer and a softmax layer are added at the top of the model to produce conditional probabilities over the class space. To avoid overfitting, dropout with a masking probability p is applied to the penultimate layer. The key idea of dropout is to randomly drop units (along with their connections) from the neural network during the training phase [19]. This output layer is calculated as follows:

$$z = \begin{cases} W_s(s \odot q) + b_s & \textit{training phase} \\ W_s([s, \tilde{s}]) + b_s & \textit{testing phase} \end{cases}, \tag{12}$$

$$y_c = \frac{exp(z_c)}{\sum_{c' \in C} exp(z_c')}, \tag{13}$$

where $q$ is the masking vector with dropout rate $p$ which is the probability of dropping a unit during training, and $C$ is the class number. In addition, a $l - 2$ norm constraint of the output weights $W$ s is imposed during training as well.

Let $\tilde{y}_c$ denotes the label of a sentence. Cross entropy loss function is given by:

$$\mathbb{L} = -\sum_{i \in N} \sum_{c \in C} \tilde{y}_c(S_i) log(y_c(S_i)), \tag{14}$$

where $\tilde{y}_c$ codes in 1-of-K schema whose dimension corresponding to the true class is 1 while all others being 0. The parameters to be determined by the model include all the weights and bias terms in the convolutional filters, the Tree-LSTM and the softmax classifier. The attention weights will be updated during the training phase. Word embeddings are fine tuned as well. Optimization is performed using the Adadelta update rule of [20], which has been shown as an effective and efficient back-propagation algorithm.

## 3 Experiments and Results

### 3.1 Classification

In this section, we evaluate the performance of the proposed model on the Stanford Sentiment Treebank [21] benchmark dataset and compare it with several state-of-the-art approaches.

The Stanford Sentiment Treebank contains about 11,800 sentences from the movie reviews. The sentences were parsed with the Stanford parser [22]. There are two subtasks: binary classification of sentences excluding neutral reviews with class distribution of 4955/4663., and fine-grained classification over five classes: very positive, positive, neutral, negative, very negative, with class distribution of 1837/3118/2237/3147/1516. Standard binarized constituency parse trees are provided for each sentence in the dataset, and each node in these trees is annotated with a sentiment label.

[23] provide a guide regarding CNN architecture and hyperparameters for practitioners who deploy CNNs for sentence classification tasks. We initialized our word representations using publicly available 300-dimensional word2vec vectors. Word representations were updated during training with a learning rate of 0.1. The DCNN has two wide convolution layers with filters whose width is 7 and 5 respectively [7] and

100 feature maps. The networks use the *tanh* non-linear function. DCNN models were trained using Adadelta with a learning rate of 0.05 and a minibatch size of 50.

The output dimension of the Tree-LSTM is set the same as the number of feature maps in order to compare Tree-LSTM representations with DCNN representations. The training batch size is set as 100.

All parameters were regularized with a per-minibatch L2 regularization strength of 10–4. The sentiment classifier was additionally regularized using dropout with a dropout rate of 0.5.

### 3.2 Result

**Table 1.** Accuracy (%) of our model and other methods from literature. The presented results are the test set accuracy of the run with the highest accuracy on the validation set.

| Method | Fine-grained | Binary |
|---|---|---|
| CNN-non-static | 48.0 | 87.2 |
| CNN-multichannel | 47.4 | 88.1 |
| DCNN | 48.5 | 86.8 |
| LSTM | 46.4 | 84.9 |
| Bidirectional LSTM | 49.1 | 87.5 |
| Tree-LSTM | 51.0 | 88.0 |
| APCNN | 50.1 | 89.9 |
| This work | 50.6 | 88.7 |

Experiment results against other methods are listed in Table 1. The performance of AP-DCNN outperforms the state-of-the-art models like BLSTM and APCNN on the fine-grained classification subtask and achieves accuracy comparable to APCNN and Tree LSTM on the binary classification subtask.

## 4 Conclusion

In the present work, a new neural semantic sentence model termed Attention Pooling DCNN has been successfully developed. We introduce Tree-LSTM to attention mechanism to model sentence. Our model is able to capture long term and syntactic information. We evaluated the learned semantic sentence representations on sentiment classification task with very satisfactory results and good performance.

**Acknowledgements.** This research was supported by National High-tech R&D Program (863 Program No 2015AA015403) and National Natural Science Foundation of China (No 61370131).

## References

1. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
2. Bengio, Y., Ducharme, R., Vincent, P., Jauvin, C.: A neural probabilistic language model. J. Mach. Learn. Res. **3**(Feb), 1137–1155 (2003)
3. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality, pp. 3111–3119 (2013)
4. Kim, Y.: Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882 (2014)
5. Socher, R., Pennington, J., Huang, E.H., Ng, A.Y., Manning, C.D.: Semi-supervised recursive autoencoders for predicting sentiment distributions. In: Association for Computational Linguistics, pp. 151–161 (2011)
6. Lawrence, S., Giles, C.L., Fong, S.: Natural language grammatical inference with recurrent neural networks. IEEE Trans. Knowl. Data Eng. **12**(1), 126–140 (2000)
7. Kalchbrenner, N., Grefenstette, E., Blunsom, P.: A convolutional neural network for modelling sentences. arXiv preprint arXiv:1404.2188 (2014)
8. Socher, R., Lin, C.C., Manning, C., Ng, A.Y.: Parsing natural scenes and natural language with recursive neural networks, pp. 129–136 (2011)
9. Socher, R., Manning, C.D., Ng, A.Y.: Learning continuous phrase representations and syntactic parsing with recursive neural networks, pp. 1–9 (2010)
10. Funahashi, K., Nakamura, Y.: Approximation of dynamical systems by continuous time recurrent neural networks. Neural Netw. **6**(6), 801–806 (1993)
11. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. IEEE Trans. Sig. Process. **45**(11), 2673–2681 (1997)
12. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
13. Graves, A., Schmidhuber, J.: Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Netw. **18**(5), 602–610 (2005)
14. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. P IEEE **86**(11), 2278–2324 (1998)
15. Er, M.J., Zhang, Y., Wang, N., Pratama, M.: Attention pooling-based convolutional neural network for sentence modelling. Inf. Sci. **373**, 388–403 (2016)
16. Tai, K.S., Socher, R., Manning, C.D.: Improved semantic representations from tree-structured long short-term memory networks. arXiv preprint arXiv:1503.00075 (2015)
17. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
18. Johnson, R., Zhang, T.: Effective use of word order for text categorization with convolutional neural networks. arXiv preprint arXiv:1412.1058 (2014)
19. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
20. Zeiler, M.D.: ADADELTA: an adaptive learning rate method. arXiv preprint arXiv:1212.5701 (2012)
21. Socher, R., Perelygin, A., Wu, J.Y., Chuang, J., Manning, C.D., Ng, A.Y., Potts, C.: Recursive deep models for semantic compositionality over a sentiment treebank, p. 1642. Citeseer (2013)
22. Klein, D., Manning, C.D.: Accurate unlexicalized parsing. In: Association for Computational Linguistics, pp. 423–430 (2003)
23. Zhang, Y., Wallace, B.: A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. arXiv preprint arXiv:1510.03820 (2015)