TreeNet: Learning Sentence Representations with Unconstrained Tree Structure

Zhou Cheng[1], **Chun Yuan**[2], **Jiancheng Li**[1] and **Haiqin Yang**[3]
[1] Department of Computer Science and Technology, Tsinghua University
[2] Graduate school at ShenZhen, Tsinghua University
[3] Department of Computing, Hang Seng Management College
{z-cheng15,ljc-15}@mails.tsinghua.edu.cn, yuanc@sz.tsinghua.edu.cn, hqyang@ieee.org

Abstract

Recursive neural network (RvNN) has been proved to be an effective and promising tool to learn sentence representations by explicitly exploiting the sentence structure. However, most existing work can only exploit simple tree structure, e.g., binary trees, or ignore the order of nodes, which yields suboptimal performance. In this paper, we proposed a novel neural network, namely TreeNet, to capture sentences structurally over the raw unconstrained constituency trees, where the number of child nodes can be arbitrary. In TreeNet, each node learns from its left sibling and right child in a bottom-up left-to-right order, thus enabling the net to learn over any tree. Furthermore, multiple soft gates and a memory cell are employed in implementing the TreeNet to determine to what extent it should learn, remember and output, which proves to be a simple and efficient mechanism for semantic synthesis. Moreover, TreeNet significantly suppresses convolutional neural networks (CNN) and Long Short-Term Memory (LSTM) with fewer parameters. It improves the classification accuracy by 2%-5% with 42% of the best CNN's parameters or 94% of standard LSTM's. Extensive experiments demonstrate TreeNet achieves the state-of-the-art performance on all four typical text classification tasks.

1 Introduction

Most neural networks for sentence representation commonly fall into one of the following classes: sequence models, convolutional models and recursive models. Recently, sequence models have been witnessed the resurge of recurrent networks in natural language processing (NLP) applications, such as text classification [Wang *et al.*, 2016; Zhang *et al.*, 2017], machine translation [Sutskever *et al.*, 2014], question answering [Wang and Nyberg, 2015] and so on. In these methods, a recurrent neural network (RNN) takes each word as the input, aggregates the word with its previous state and finally outputs its compositional result over the whole sentence. The compositional result as a fixed-length vector contains rich semantic information and is used for the subsequent NLP tasks.

However, in natural languages understanding, these recurrent networks, including LSTM [Hochreiter and Schmidhuber, 1997], handle the word sequences in a flat sequential or bidirectional sequential way, which means they directly ignore the structural information inherent in sentences. Convolutional models, which combine multiple filters and pooling operations, also neglect the language structure and use a large amount of parameters to boost their performance [Kim, 2014].

In contrast, recursive models as counterparts explicitly represent the recursive structure inherent in natural language. In [Socher *et al.*, 2011], a RvNN model is employed to comprehend sentences and scene. Furthermore, a dedicated dataset, Stanford Sentiment Dataset (SST) [Socher *et al.*, 2013], is published with each sentence in a binary tree format. Apart from sentence-level labels, each inner node in SST is given an additional supervision signal indicating nodes' sentiment. The extra information has proved to be helpful when training a recursive model [Dong *et al.*, 2014; Tai *et al.*, 2015; Zhu *et al.*, 2015]. It is worth noting that these recursive models outperform the LSTM on the SST dataset and attract more attention from researchers to design more complicated recursive compositional functions [Liu *et al.*, 2017a; 2017b; Teng and Zhang, 2017].

Albeit the improvement on performance, the recursive networks working on the SST are highly dependent on the binary tree or binarized constituency tree, which are different from the original linguistic constituency tree. Even though the Child-Sum TreeLSTM in [Tai *et al.*, 2015] supports to process arbitrary trees, it doesn't fully exploit the order of child nodes as these children are simply summed to represent themselves. In a word, to the best of our knowledge, there exists no previous work that exploited sentence structure fully, applied to the raw constituency tree and achieved outstanding results.

In this paper, we proposed a tree-based model to make full use of sentences' structural information. The proposed TreeNet firstly utilises a recurrent operation to learn the representation of child nodes and then employs a recursive operation to model the relations between a parent node and its newly learnt children's representation. Specifically, an inner node learns from its left sibling node and right child node in a left-to-right bottom-up order. It is the partial order of computing along with the proposed compositional function

that makes the root node of one sentence absorb from its descendants all related semantic information effectively and efficiently. As the child nodes are processed in a recurrent manner, TreeNet naturally supports to learn over any tree, which makes it distinct from previous recursive methods. Furthermore, multiple soft gates and a memory cell are used in implementing a TreeNet to reflect the importance of different constituents.

We evaluate our method on four typical text classification tasks and demonstrate that the proposed TreeNet significantly outperforms the state-of-the-art models over all tasks without bells and whistles. The experiments show that our method is more expressive due to its processing sentences in a more natural way. Moreover, compared to the CNN-based and LSTM-based methods, TreeNet achieves much better performance with fewer parameters.

In summary, the proposed TreeNet contains the following merits:

- It can learn an arbitrary tree recursively and naturally from any sentence datasets. It explicitly exploits the sentences' structural information provided by a parser.
- In learning sentence representation, the dedicated gate-memory based design enables TreeNet to model sentences efficiently and flexibly with fewer parameters. The amount of parameters in TreeNet is 42% of the best CNN's and 94% of a basic LSTM's.
- Generally and more importantly, TreeNet provides a novel paradigm for processing and modelling sentences.

2 Related Work

It is natural to represent a sentence as a tree and lots of efforts have been put into natural language processing tasks to solve them in a recursive way. A core problem is how to design or learn a compositional function to better summarise descendants' information [Socher *et al.*, 2011; 2013; Tai *et al.*, 2015; Liu *et al.*, 2017b; Teng and Zhang, 2017]. Recursive Neural Network [Socher *et al.*, 2011] learns a weighted linear combination of the left and the right child vector as their parent. Recursive Neural Tensor Network [Socher *et al.*, 2013] employs tensors to model intersections between different dimensions of child vectors. [Dong *et al.*, 2014] uses a composition pool to decide the best output dynamically, where each composition function in the pool is separately scored in feedforward. All of these works assume the input is a binary tree and their experiments are conducted on binary trees or the binarized constituency trees.

Moreover, TreeLSTM [Tai *et al.*, 2015], a modified LSTM, is employed as a composition function to learn the state from children's state. Child-Sum TreeLSTM is an extension of TreeLSTM to compute the parent node via a standard LSTM and the previous state of a node via sum of children's states (see Eq. (2) (7)). N-ary TreeLSTM is another extension to apply a matrix multiplication over children's states to represent the previous state (see Eq. (9)-(13)). By encoding nodes and their children with LSTM, TreeLSTM learns a more complicated representation and outperforms its LSTM baseline. However, the Child-Sum TreeLSTM processes children's states in an unordered way while N-ary TreeLSTM in a position-sensitive way. That is, they do neither fully exploit nodes' order nor work on more general tree structure.

3 Background

In this section, we describe the vanilla recurrent networks and LSTM as well. We will briefly discuss how they inspire the implementation of the TreeNet for learning sentence representations.

3.1 Vanilla Recurrent Networks

Recurrent neural networks (RNNs) apply a state transition function to a hidden state vector h and an input vector x to produce a sequence representation of arbitrary length. At each step t, the hidden state h_t is the output of a function which takes as input the output of previous step h_{t-1} and the received signal x_t, specifically the word vector of t-th word in one sentence [Elman, 1990; Tomas, 2012].

Commonly, the state transition function is an affine transformation followed by an element-wise activation function such as the hyperbolic tangent. And the hidden state $h_t \in \mathbb{R}^d$ denotes the d-dimensional distributed representation of a sequence of observed tokens up to step t.

$$h_t = \tanh(Wx_t + Uh_{t-1} + b) \tag{1}$$

The final output of RNN is commonly used as sentence representations and fed into a classifier to predict the sequence attributes. However, RNN's performance suffers from *the exploding and the vanishing gradient* problems when training with long sequences like complex sentences, documents and speech signal.[Bengio *et al.*, 1994; Hochreiter, 1998].

3.2 LSTM

To remedy RNN's drawback, LSTM proposed in [Hochreiter and Schmidhuber, 1997] utilises a *memory cell* with three *gates* to maintain the state of an observed sequence. In LSTM, the previous state along with the current input is selectively incorporated into a novel state, which is then used as an inner representation of all observations. Here we describe the version used in [Zaremba and Sutskever, 2014] and mentioned in [Tai *et al.*, 2015].

At time step t, LSTM firstly performs an affine transformation on the input x_t and previous output h_{t-1}, and then computes its input gate i_t, forget gate f_t and output gate o_t with a logistic sigmoid function. The candidate representation $\widetilde{c}$ remains the same as RNN's. All of them except the input are d-dimensional vectors, where d is the dimension of the memory cell. Besides, gates are limited in a range of [0,1]. In order to update the memory cell, the forget gate and input gate are multiplied with the previous cell c_{t-1} and the candidate $\widetilde{c}$ respectively. LSTM generates its current output h_t by applying a non-linearity function to the updated cell c_t and then multiplying it with the output gate. All the LSTM

transition functions are as follows.

$$
\begin{aligned}
i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i), \\
o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o), \\
f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f), \\
\widetilde{c} &= \tanh(W_u x_t + U_u h_{t-1} + b_u), \\
c_t &= f_t \odot c_{t-1} + i_t \odot \widetilde{c}, \\
h_t &= o_t \odot \tanh(c_t),
\end{aligned}
\tag{2}
$$

where σ denotes a logistic sigmoid function and $\odot$ is the element-wise multiplication and x_t is the input at the current time step.

Intuitively, by multiplying the forget gate with the previous cell state c_{t-1}, LSTM's cell absorbs partial history information with current input x_t. Thus it makes sense that LSTM learns a multi-scale dependency over long periods of time. The memory cell and gate mechanism used in LSTM's input, hidden state and output finds to be effective in representing sequences and preventing gradient-related problems. Therefore a memory cell and two gates are also employed our method to preserve one node's inner state and decide which constituents are more important.

4 Methodology

In this section, we firstly discuss why a constituency tree, the input of TreeNet, is better than its corresponding sequence and binary tree. Then the general TreeNet is present to explain how TreeNet learns over a tree. In addition, we state why it's a recurrent and recursive method. Next, we formulate a concrete gate-memory based TreeNet for modelling sentences, which is also the TreeNet used in the experiments. The last part of this section gives a brief introduction to the output layer and loss function in text classification tasks.

4.1 Constituency Tree

Although sentences exist in a sequential format in computers and real world, linguists state human beings interpret a sentence in a tree-based way, specifically the *constituency tree*. In this paper, constituency tree is taken as the input format of one sentence while the binary tree and its original sequence are two formats which we think are not most suitable for models to learn.

To demonstrate the difference among these three formats, an ordinary sentence *You won't like Rogers, but you will quickly recognise him.*[1] is chosen from the SST dataset. Figure 1 displays its constituency tree and one can easily retrieve the corresponding binary tree from SST's website. From Figure 1, we find that one sentence is teared semantically into chunks, phrases and semantic snippets in its constituency tree. Furthermore, as the figure shows, a constituency tree is more shallow than its counterpart binary tree[2]. In other words, a constituency tree represents a sentence in a more compact and semantic format than its sequence and binary tree.

[1]https://nlp.stanford.edu/sentiment/treebank.html, figure id: 000092.

[2]The depth of the corresponding binary tree is 8.

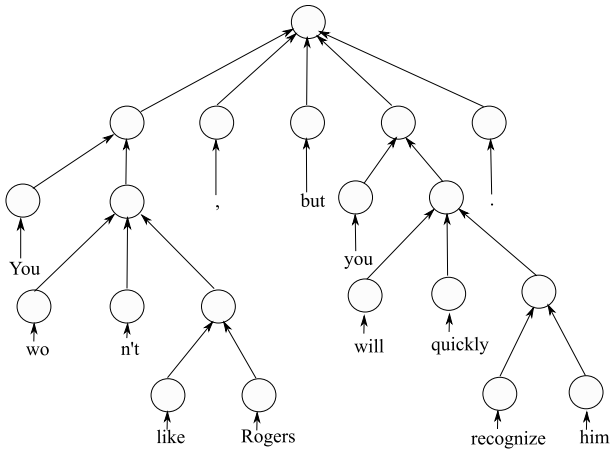

Figure 1: The constituency tree of "*You won't like Rogers, but you will quickly recognise him.*" (SST#000092). Note that the original constituency tree exists some nodes holding only one child, and an operation of removing those nodes and linking their children to their parents is performed in this figure and experiments.

In order to apply a preset compositional function, most recursion-based work manually smashes the constituency tree into a binary tree to avoid the problem that the number of child nodes in a constituency tree is variable. As a consequence, they either sacrifice the raw constituency tree or not fully exploit it, such as the Child-Sum Tree-LSTM in [Tai *et al.*, 2015]. TreeNet addresses this dilemma in a natural way.

4.2 General TreeNet

The general TreeNet consists of two parts: token encoder and semantic compositor. In a constituency tree, tokens, such as words and punctuation, are the basic elements and the leaf nodes. Because the dimension of word vectors is generally different from sentence vectors', the token encoder functions as a semantic transition part between words and sentences. Then the semantic compositor learns an inner node's representation over the constituency tree in a partial order.

- **Token Encoder**: turns leaf nodes (word embeddings) into sentences' semantic space.
- **Compositor**: learns current node's representation by incorporating the node's left sibling and the right child.

Figure 2 illustrates the dependency graph of computing over a constituency tree. The dependency graph forms a partial order which we think conforms to the cognitive rules intuitively.

For each sentence, a d-dimensional distributed vector is used as its semantic representation whereas the token may be represented as a one-hot vector, a random distributed vector in any dimension or a pre-trained vector derived from other works such as GloVe [Pennington *et al.*, 2014]. Therefore it is necessary for TreeNet to utilise the token encoder to learn a transition function, which encodes tokens' distributed representations into sentences' representations.

$$
s_w = Encoder(w_{emb}), \tag{3}
$$

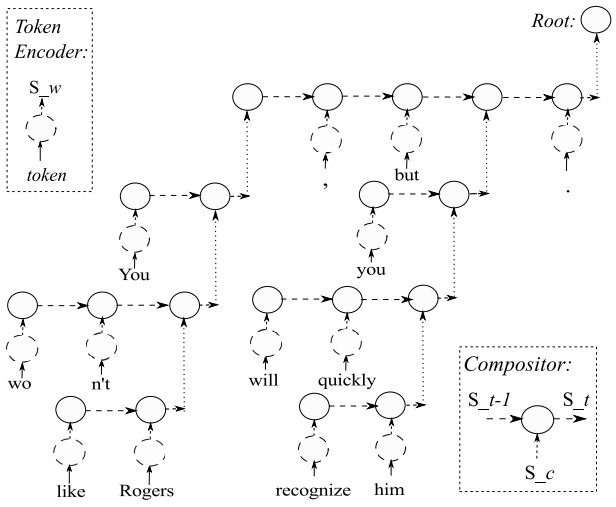

Figure 2: The computation procedure over a constituency tree. Firstly, tokens are encoded from the word space to the sentence space. Secondly, inner nodes with the same parent are computed from left to right sequentially. Meanwhile, they incorporate into themselves the right child's representation, which represents all of descendants.

, where w_{emb} denotes the word vector of w-th word and s_w is its corresponding vector in sentence's semantic space. The $Encoder$ stands for any neural networks.

Different from the leaf nodes, each inner node owns one or more siblings and children. Since the left inner node has no left sibling, a zero-initialised vector is used during computing. For an inner node with multiple children, the output of the right child is used as the representation of its descendants. All child nodes of the same parent are processed sequentially from left to right, in which way a person understands one sentence. The formula reads as follows,

$$s_t = Compositor(s_{t-1}, s_c), \tag{4}$$

where s_c denotes current node t's unique child (one token's representation s_w) or right child representing all descendants, s_{t-1} denotes the output of its left sibling and the $Compositor$ function can be any networks learning how to incorporate them.

In TreeNet, the compositor function with the computation procedure plays an essential role in comprehending sentences or partial sentences. From the viewpoint of siblings, the compositor handles them sequentially like a recurrent net as it assumes the child is its input and the left sibling is its previous state. On the side of the relation between parent nodes and children, parents are updated in a recursive way as they apply the compositor recursively to their left siblings and right children. We believe this characteristic of TreeNet is attributed to the elegant computation procedure and the intrinsic structure of language.

4.3 Gate-Memory TreeNet

Although there exist a large number of candidate networks (e.g. various Multilayer Perceptron) to implement the word encoder and compositor in a general TreeNet, we introduce a simple gate and memory based TreeNet, namely Gate-Memory TreeNet. In the Gate-Memory TreeNet, a memory cell is utilised to hold all observations for each node and multiple gates for the input and output.

Gate-Memory Token Encoder

Gate-memory token encoder employs a dedicated gate-based network to learn one token's sentence representation. To be specific, the gate-based token encoder learns two gates to determine what should be integrated into its memory cell and what should be output as its sentence representation respectively. Basically, an affine transformation is learnt to map word vectors to sentence vectors. The encoder is defined as follows:

$$i_w = \sigma(W_{wi} w_{emb} + b_{wi}), \tag{5}$$
$$o_w = \sigma(W_{wo} w_{emb} + b_{wo}), \tag{6}$$
$$u_w = \tanh(W_{wu} w_{emb} + b_{wu}), \tag{7}$$
$$c_w = i_w \odot u_w, \tag{8}$$
$$h_w = o_w \odot \tanh(c_w) \tag{9}$$

where W_{wi}, W_{wo} and W_{wu} denote the weights used to compute the input gate i_w, output gate o_w and candidate sentence vector u_w. The current token's embedding is w_{emb}. The logistic sigmoid function σ constrains results of matrix operations in a range of [0,1]. Therefore, it is two soft gates that are involved into learning the memory cell c_w and the output h_w. In addition, i_w, o_w, u_w, c_w and h_w keep the same dimensionality as sentence representations', except that the dimension of w_{emb} may be defined by pre-trained word vectors or arbitrary random vectors.

It's worth noting that the memory cell c_w and output h_w are used like a regular inner node's memory cell and output in the following computation.

Gate-Memory Compositor

At inner node t, its left sibling (h_s, c_s) is defined as follows,

$$(h_s,\ c_s) = \begin{cases} (0,\ 0) & \text{if } t \text{ is the first child} \\ (h_{t-1},\ c_{t-1}) & \text{otherwise} \end{cases} \tag{10}$$

and its child (h_c, c_c) is:

$$(h_c,\ c_c) = \begin{cases} (h_{rc},\ c_{rc}) & \text{if has multiple children} \\ (h_w,\ c_w) & \text{otherwise, a token child} \end{cases} \tag{11}$$

where (h_{rc},c_{rc}) denotes the memory cell and output of its right child. And then the gate-memory compositor is defined as follows,

$$i_s = \sigma(W_{ss} h_s + W_{sc} h_c + b_s), \tag{12}$$
$$i_c = \sigma(W_{cs} h_s + W_{cc} h_c + b_c), \tag{13}$$
$$o_t = \sigma(W_{so} h_s + W_{co} h_c + b_o), \tag{14}$$
$$c_t = i_s \odot c_s + i_c \odot c_c, \tag{15}$$
$$h_t = o_t \odot \tanh(c_t), \tag{16}$$

where h_s denotes the output of left sibling and h_c is the output of its child node or right child, c_s and c_c are the memory cells of them respectively. Two soft gates i_s, i_c determine to what extent the two cells contribute to the current memory cell c_t. Then the output h_t is element-wise product of the output gate o_t and the memory cell with a non-linearity hyperbolic tangent function.

4.4 Output Layer

For text classification, the root node's output h_{root} is the whole representation of the input text. And then it is passed to a softmax classifier to predict the probability of corresponding classes.

$$\hat{p}(y|s) = softmax(W_{root}h_{root} + b_{root}) \quad (17)$$

The parameters of the model are trained to minimise the cross-entropy between the prediction and true label distributions. The loss is calculated as a regularised sum:

$$J(\theta) = -\frac{1}{n}\sum_{i=1}^{n} t_i log(y_i) + \lambda\|\theta\|^2 \quad (18)$$

where $t \in \mathbb{R}^n$ is the one-hot vector of ground truth, $y \in \mathbb{R}^n$ is the predicted probability for each class, n is the number of target classes and λ is an L2 regularisation hyper-parameter.

5 Experiments

5.1 Datasets

The experiments are conducted on four widely used benchmark datasets for text classification, where accuracy is applied to measure the classification performance. The description of each dataset is given as follows.

- **MR**: In the Movie Reviews dataset, each sentence is selected from one movie review with an assigned positive or negative label about reviewer's attitude.[Pang and Lee, 2005] [3]
- **Subj**: The Subjectivity dataset is to classify a sentence as being subjective or objective.[Pang and Lee, 2004]
- **TREC**: TREC QC consists of six typical question types (person, location, numeric information, description, entity and abbreviation), where the task is to assign each sentence to its corresponding question type.[Li and Roth, 2002][4]
- **CR**: The Customer Reviews dataset consists of five products (two digital camera, one cellular phone, one mp3 player and one dvd player), where the task is to classify each customer review as a positive or negative review.[Hu and Liu, 2004][5]

To verify the performance of TreeNet (the Gate-Memory TreeNet), we select the above four datasets based on the following two reasons: First, the performance of the compared models is directly reflected by the classification accuracy of sentences. The selected datasets do not have any extra supervision information on the intermediate nodes because obtaining these auxiliary labels is time-consuming. Second, the average sentence length is suitable because a dataset with too many phrases or short sentences cannot characterize the complexity of language in all aspects. Table 1 provides statistics of these datasets.

[3]https://www.cs.cornell.edu/people/pabo/movie-review-data/, the Subj (Subjectivity) dataset is also provided in this website.

[4]http://cogcomp.cs.illinois.edu/Data/QA/QC/

[5]http://www.cs.uic.edu/ /liub/FBS/sentiment-analysis.html

Dataset	N	$Test$	$\|W\|$	$\|W_{init}\|$	c	l
MR	10662	CV	18765	16448	2	21
Subj	10000	CV	22377	20832	2	23
TREC	5952	500	9759	9569	6	10
CR	3775	CV	5628	5480	2	19

Table 1: Summary statistics of the datasets. N: Total number of sentences excluding the test sentences if provided. $Test$: The number of test sentences. W: Number of tokens generated by the Stanford parser. W_{init}: Number of tokens initialised by pre-trained word vectors. c: Number of classes (opinion polarity, sentiment types or question types). l: average sentence length. CV means there is no standard train/dev/test split and a random 10-fold CV is used.

5.2 Training

In the experiments, the input sentence is tokenised and parsed by Stanford Parser[6], which generates a bucket of tokens and a constituency tree. In experiments $TreeNet_{\text{-}glove}$, the word embeddings are built on all of generated tokens and initialised from pre-trained word embeddings GloVe [Pennington *et al.*, 2014] and tokens not present in GloVe are sampled from a uniform distribution in [-0.05, 0.05]. We did not fine-tune the pre-trained word embeddings. We reported the results of $TreeNet$ (without GloVe) by initialising all tokens from the uniform distribution in [-0.05, 0.05]. For the constituency tree, every inner node with only one child is dropped and its child is linked to its parent, which will compress a tree with a proper depth without diminishing useful information. Thus all inner nodes have at least two children (subtrees) or a token (leaf) as the unique child.

In the experiments without GloVe, namely $TreeNet$, the dimension of word embeddings is 100 and dimension of sentence representations is 50. In the experiments with GloVe,namely $TreeNet_{\text{-}glove}$, one word embedding is a 300-dimensional vector and a sentence representation is a 100-dimensional vector. In all these experiments, the model parameters are optimised through stochastic gradient descent over shuffled mini-batches with the Adam [Kingma and Ba, 2014] and batch size of 25. In order to get best performance in experiments, we conducted a grid search on the learning rate in the range of [1e-2, 1e-5] and L2 regularisation strength in the set of (1e-3, 1e-4, 1e-5, 0).

5.3 Results and Analysis

The compared models can be grouped into four types: recursive models, CNNs, LSTMs and others like n-gram based methods. Table 2 reports the classification results compared with those models. From the comparison, we find TreeNet consistently outperforms LSTM, RvNNs (DC-TreeLSTM, T-LSTM and AdaHT-LSTM) and CNNs in a large margin and significantly outperforms the state-of-the-art methods on all four datasets.

In order to take TreeNet as a general network in modelling sentences, we prudently conducted comparative evaluations among CNN and LSTM with the same hyper-parameters on MR dataset. In this experiment, they all used pre-trained word

[6]http://nlp.stanford.edu/software/lex-parser.shtml

Models	MR	Subj	TREC	CR
NBSVM	79.4	93.2	-	81.8
MNB	79.0	93.6	-	80.0
combine-skip+NB	80.4	93.6	-	81.3
AdaSent	83.1	95.5	92.4	86.3
DC-TreeLSTM	81.7	93.7	93.8	-
Tree-LSTM	78.7	91.0	91.6	-
AdaHT-LSTM-CM	81.9	94.1	-	-
CNN-no-static	81.5	93.4	93.6	84.3
CNN-Ana	81.02	93.66	91.37	84.65
DSCNN	81.5	93.2	95.4	-
BLSTM	80.0	92.1	93.0	-
BLSTM-Att	81.0	93.5	93.0	-
BLSTM-2DCNN	82.3	94.0	96.1	-
LSTM	77.5	91.5	88.6	75.7
LSTM$_{-glove}$	80.7	93.4	93.6	83.8
$TreeNet$	79.8	92.0	91.6	77.4
$TreeNet_{-glove}$	**83.6**	**95.9**	**96.1**	**88.4**

Table 2: Classification results. LSTM and LSTM$_{-glove}$ are our baselines with default parameters. **NBSVM**: [Wang and Manning, 2012], **MNB**:[Wang and Manning, 2012], **combine-skip+NB**: [Kiros *et al.*, 2015], **AdaSent**: [Zhao *et al.*, 2015]; **DC-TreeLSTM**: [Liu *et al.*, 2017b]. **Tree-LSTM**, **AdaHT-LSTM-CM**: [Liu *et al.*, 2017a]. **CNN-no-static**: [Kim, 2014], **CNN-Ana**: [Zhang and Wallace, 2015], **DSCNN**: [Zhang *et al.*, 2016]. **BLSTM**, **BLSTM-Att**, **BLSTM-2DCNN**: [Zhou *et al.*, 2016]

embeddings. The CNN architecture and result is referred to [Kim, 2014], which reports the highest performance among single CNN models with word2vec, where it stated utilising word2vec yields better performance than GloVe. For LSTM and TreeNet, we used GloVe to initialise the word embeddings. We reproduced LSTM on MR dataset with hidden size of 100, learning rate of 0.001 and L2 regularisation strength of 0. Importantly, TreeNet keeps the exactly same configuration as LSTM's in the controlled experiment. During training, a sentence is fed into LSTM and TreeNet simultaneously. Table 3 gives a comparison about learnable parameters and performance. Figure 3 plots the training loss and test accuracy under that condition. In the figure, LSTM and TreeNet achieve their best test performance at the 7th epoch (133*500 iterations) while LSTM has a smaller training loss and weaker performance. In other words, TreeNet achieves better performance and generalisation with fewer parameters.

5.4 Discussions

It is noted that by fully exploiting the structural information embedded in the sentences, our proposed TreeNet can outperform previous method. Moreover, the well-designed gate and memory mechanisms make it flexible to decide what should be learnt, remembered and output.

However, several issues are worthy of further exploration. First, the sentence structure provided by parsers is not always correct. Though the error rate of parsers is low, the impact is not negligible. It is valuable to explore what extent it will penalise the TreeNet. Second, we do not fine-tune the pre-trained word embeddings. It is interesting to test the performance with fine-tuned results. Lastly, POS (Part-of-Speech) tags in constituency trees are not exploited in the current TreeNet. It is worthy to include them in TreeNet to learn better gates.

Models	W_{emb}	$\|h\|$	#Params	Acc.@MR
CNN [Kim, 2014]	300	300	∼360k	81.5
LSTM	300	100	∼160k	80.7
TreeNet(ours)	300	100	∼**150k**	**82.1**

Table 3: Comparison on the number of parameters (#Params) and accuracy over MR (Acc.@MR). W_{emb} is the dimensionality of word embedding and $|h|$ denotes hidden states or channels. LSTM and TreeNet are trained with the same hyper-parameters.

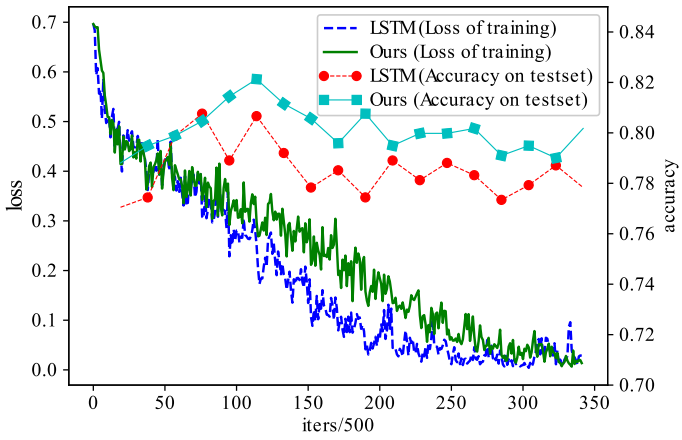

Figure 3: Comparison between LSTM and TreeNet with the same hyper-parameters and word embeddings. TreeNet achieves better and more robust performance.

6 Conclusion

This paper introduces a tree-based network to improve sentences' semantic representations over the constituency trees. The proposed model can process unconstrained tree-structure input naturally to learn sentences' structure information, which makes it different from previous tree-based works and achieves significant performance improvement. The experiments are conducted on four sentence classification tasks and experimental results demonstrate that the proposed TreeNet achieves the state-of-the-art accuracy among all methods without bells and whistles. To better understanding its effectiveness and efficiency, this work also conduct a sensitivity analysis on the Movie Review dataset and finds that TreeNet achieves better generalisation and performance than LSTM with fewer parameters.

Acknowledgments

This work is supported by the National High Technology Research and Development Plan (863 Plan) under Grant No. 2015AA015803, the NSFC project under Grant No. U1433112, and the Joint Research Center of Tencent & Tsinghua University. The work described in this paper was partially supported by the Research Grants Council of the

Hong Kong Special Administrative Region, China (Project No.UGC/IDS14/16).

References

[Bengio *et al.*, 1994] Yoshua Bengio, Patrice Y. Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5 2:157–66, 1994.

[Dong *et al.*, 2014] Li Dong, Furu Wei, Ming Zhou, and Ke Xu. Adaptive multi-compositionality for recursive neural models with applications to sentiment analysis. In *AAAI*, 2014.

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

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Júrgen Schmidhuber. Long short-term memory. *Neural computation*, 9 8:1735–80, 1997.

[Hochreiter, 1998] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6:107–116, 1998.

[Hu and Liu, 2004] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *KDD*, 2004.

[Kim, 2014] Yoon Kim. Convolutional neural networks for sentence classification. In *EMNLP*, 2014.

[Kingma and Ba, 2014] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

[Li and Roth, 2002] Xin Li and Dan Roth. Learning question classifiers. In *COLING*, 2002.

[Liu *et al.*, 2017a] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Adaptive semantic compositionality for sentence modelling. In *IJCAI*, 2017.

[Liu *et al.*, 2017b] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Dynamic compositional neural networks over tree structure. In *IJCAI*, 2017.

[Pang and Lee, 2004] Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *ACL*, 2004.

[Pang and Lee, 2005] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*, 2005.

[Pennington *et al.*, 2014] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *EMNLP*, 2014.

[Socher *et al.*, 2011] Richard Socher, Cliff Chiung-Yu Lin, Andrew Y. Ng, and Christopher D. Manning. Parsing natural scenes and natural language with recursive neural networks. In *ICML*, 2011.

[Socher *et al.*, 2013] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, 2013.

[Sutskever *et al.*, 2014] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014.

[Tai *et al.*, 2015] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *ACL*, 2015.

[Teng and Zhang, 2017] Zhiyang Teng and Yue Zhang. Head-lexicalized bidirectional tree lstms. *TACL*, 5:163–177, 2017.

[Tomas, 2012] Mikolov Tomas. *Statistical language models based on neural networks*. PhD thesis, Brno University of Technology, 2012.

[Wang and Manning, 2012] Sida Wang and Christopher D Manning. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 90–94. Association for Computational Linguistics, 2012.

[Wang and Nyberg, 2015] Di Wang and Eric Nyberg. A long short-term memory model for answer sentence selection in question answering. In *ACL*, 2015.

[Wang *et al.*, 2016] Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. Attention-based lstm for aspect-level sentiment classification. In *EMNLP*, 2016.

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

[Zhang and Wallace, 2015] Ye Zhang and Byron C. Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *CoRR*, abs/1510.03820, 2015.

[Zhang *et al.*, 2016] Rui Zhang, Honglak Lee, and Dragomir R. Radev. Dependency sensitive convolutional neural networks for modeling sentences and documents. In *HLT-NAACL*, 2016.

[Zhang *et al.*, 2017] Honglun Zhang, Liqiang Xiao, Yongkun Wang, and Yaohui Jin. A generalized recurrent neural architecture for text classification with multi-task learning. In *IJCAI*, 2017.

[Zhao *et al.*, 2015] Han Zhao, Zhengdong Lu, and Pascal Poupart. Self-adaptive hierarchical sentence model. In *IJCAI*, pages 4069–4076, 2015.

[Zhou *et al.*, 2016] Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. In *COLING*, 2016.

[Zhu *et al.*, 2015] Xiao-Dan Zhu, Parinaz Sobhani, and Hongyu Guo. Long short-term memory over recursive structures. In *ICML*, 2015.