JMLR: Workshop and Conference Proceedings 63:17–32, 2016 ACML 2016

Long Short-term Memory Network over Rhetorical Structure Theory for Sentence-level Sentiment Analysis

Xianghua Fu FUXH@SZU.EDU.CN
Wangwang Liu TOCHANGE@163.COM
Yingying Xu YINGYINGYULIA@FOXMAIL.COM
Chong Yu YUCHONG1001@163.COM
Ting Wang WANGTINGWTC@163.COM
College of Computer Science and Software Engineering, Shenzhen University, Guangdong China

Editors: Robert J. Durrant and Kee-Eung Kim

Abstract

Using deep learning models to solve sentiment analysis of sentences is still a challenging task. Long short-term memory (LSTM) network solves the gradient disappeared problem existed in recurrent neural network (RNN), but LSTM structure is linear chain-structure that can't capture text structure information. Afterwards, Tree-LSTM is proposed, which uses LSTM forget gate to skip sub-trees that have little effect on the results to get good performance. It illustrates that the chain-structured LSTM more strongly depends on text structure. However, Tree-LSTM can't clearly figure out which sub-trees are important and which sub-trees have little effect. We propose a simple model which uses Rhetorical Structure Theory (RST) for text parsing. By building LSTM network on RST parse structure, we make full use of LSTM structural characteristics to automatically enhance the nucleus information and filter the satellite information of text. Furthermore, this approach can make the representations concerning the relations between segments of text, which can improve text semantic representations. Experiment results show that this method not only has higher classification accuracy, but also trains quickly.

Keywords: LSTM, Rhetorical Structure Theory, sentiment analysis

1. Introduction

Sentiment analysis is an influential research field and it focuses on subjective text analysis and processing. According to different granularity, sentiment analysis tasks can be divided into chapter level, paragraph level, sentence level, word level or even mixed. In this paper, we focus on the task of sentence-level sentiment analysis.

In literature, traditional machine learning methods often use Native Bayes, support vector machines (SVM) Koppel and Schler (2006) and Maximum Entropy to build sentiment classifier following Pang et al. (2002). These three methods need to design features by handcraft Paltoglou and Thelwall (2010) or learn discriminate features form data. Therefore, that feature selection is good or bad has great influence on the classification results. Later, deep learning method makes such a significant effect in image processing field Lee et al. (2009); Hinton and Salakhutdinov (2006) and speech recognition field Huang and Kingsbury (2013) that researchers further make it applied in natural language processing (NLP) field, such as

machine translation Cho et al. (2014) and text classification. When we use deep Learning methods, self-learning feature take place of manual selection.

However, text sentiment classification task yet need be more deeply studied. In 2013, Google proposed word2vec tool Mikolov et al. (2013b,a) for training word vector. Unlike traditional word representation, the word vector is continuous and dense. It also can express "distance" concept and capture semantic feature and syntax of words. However, because the word vector can only represents a single word, phrase and sentence representations need consider the semantic combination problem. Which methods can be used for semantic combination? In recent years, two types of deep learning models, Recursive Auto-encoder (RAE) and RNN, have been proposed to conduct semantic combination for phrase and sentence representations. These two models achieve good results, but there are some flaws. The RAE method tends to generate deep parse trees and has high training complexity. RNN, another common semantic combination model, is developed into the RNN network with LSTM unit Graves (1997) which handles text as temporal sequence. Although LSTM has observed promising result, many tasks are inherently related to structure which are more complicated than sequence. To improve feature representations of sentences, text structure still needs to dig.

To make full use of text structure and learn sufficient feature representations, the analytical tool RST can be used. Although the best accuracy of binary sentiment classification of RST method is only about seventy percent Hogenboom et al. (2015), RST can describe the relations between internal parts of text. We can use RST relations to parse a text. On the basis of the fact, we propose RST-LSTM model which conduct LSTM on the RST parse tree. RST-LSTM uses the forget components of segments to automatically enhance the nucleus information and filter the satellite information, so semantic feature representations of text will be more sufficient and more accurate. This model is trained on Stanford Sentiment Treebank dataset Socher et al. (2013). Experimental results show that this method not only has a higher classification accuracy than the state-of-the-art methods Tai et al. (2015); Socher et al. (2013, 2012, 2011b), but also trains quickly.

2. Related Works

2.1. Sentiment Analysis

The improvement of sentiment analysis tasks is mainly at the step of text feature learning. Recently, neural network methods to learn continuous text representations have been widely used because of their high performance. These methods can be divided into two directions. One is devoted to the research of word embedding. High quality word embedding is the basis of natural language processing task. Mikolov et al. (2013b) first discovers that RNN can learn the analogy relationship between words (such as apple-apples cat-cats, man-woman king-queen). Levy and Goldberg (2014) learns word embedding by using dependency syntax path as context of words on the skip-gram model Mikolov et al. (2013c). The other is devoted to the research of semantic composition Mitchell and Lapata (2010). Make sentence-level sentiment analysis as an example: sentence representations can be seen as semantic combination of all the words in sentence. In this paper, the sentence-level sentiment analysis task mainly researches how to effectively get the better semantic combination of words.

Furthermore, in the process of learning text representation, text structure may hold valuable information. Its naturally that we combine words according to text structure.

2.2. syntax parse tree

Common syntax parse trees include dependency tree and constituent tree. Dependency tree can directly handle the relations between words of sentences. Each node represents a word in sentences. In constituent tree, leaf nodes represent all words composed of sentences, non-leaf nodes represent generated phrases. In recent years, the researches of sentiment analysis based on RST gain more attention Hogenboom et al. (2015); Chenlo et al. (2014); Zhou et al. (2011). RST is a popular discourse analysis framework, which can describe the relations between internal parts of text. RST relations split text into rhetorically related segments that possibly can be split too, so it will generate a hierarchical rhetorical structure. Each segment is a nucleus or a satellite. But previous researches on RST are small-scale text sentiment classification. Recently, some researchers put forward RST tree that considers leaf nodes and entire RST hierarchy structure Hogenboom et al. (2015) to improve classification accuracy. But Chenlo et al. (2014) calculated emotional polarity values of micro-blog comments by simple weighted method of different RST tree segments. Tu et al. (2013) also creates translation framework based on RST tree. These works are carried out by directly weighted method, instead of automatic learning text semantic representation. Surdeanu et al. (2015) propose two RST analysis tools based on dependency and constituent syntax on the basis of previous RST parsers Feng and Hirst (2012); Soricut and Marcu (2003). Although the two parsers have a good effect, but they have not been effectively applied. The researches of RST are still in the shallow stage.

2.3. semantic combination model

Three main semantic combination models are bag of words model, RAE and RNN. Bag of words model gets sentence representation by simply averaging each word embedding existed in text sequences. The model is more effective for long text, but the drawback is losing word order information. RAE usually needs a topology structure, such as syntax trees. Socher et al. (2011a) obtains the entire sequence representation by constantly recursive composition. RNN handles text as temporal sequences. Current state is affected by the last step and will affect the next step. Finally, we get the entire sequences representations by considering current input and last step output at any time steps. Later, RNN with LSTM unit is explicitly designed for avoiding the gradient disappeared problem. Afterwards, Many variants of LSTM are proposed especially Tree-LSTM network Tai et al. (2015); Zhu et al. (2015). Although Tree-LSTM has observed promising result, there are many problems that remain to solve. Tree-LSTM uses LSTM forget unit to skip the sub-trees which have little effect on the results to get good performance. But we can't clearly figure out which sub-trees are important and which sub-trees have little effect on sentiment analysis. We build LSTM network based on analytical results of these two RST parsers Surdeanu et al. (2015) which are based on dependency and constituent syntax. Combining LSTM and RST will help to break their own limitations.

3. Methods

3.1. LSTM and Tree-LSTM

Recent years have seen a revival of LSTM Graves (1997). LSTM which introduces a special memory cell can solve the gradient disappeared problem. The memory cell is composed of three gates and a cell unit. Gates generally use sigmoid activation function and the cell unit often uses tanh transformation. The output values of sigmoid function is between 0 and 1, where 1 means completely carrying previous information and 0 means completely abandoning. LSTM adds or removes information to update the cell state through gates. LSTM model diagram is shown in Fig. 1. Equations of LSTM are expressed as follows:

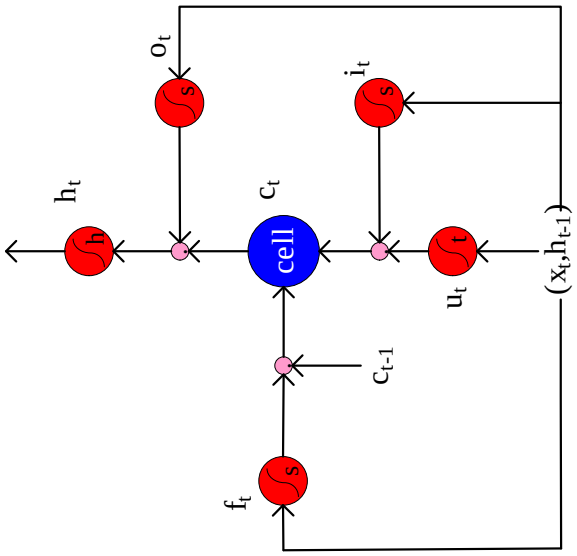


Figure 1: LSTM network.

Input gate:

$$i_t = \sigma(W^{(i)}x_t + U^{(i)}h_{t-1} + b^{(i)}) \tag{1}$$

Forget gate:

$$f_t = \sigma(W^{(f)}x_t + U^{(f)}h_{t-1} + b^{(f)}) \tag{2}$$

Output gate:

$$o_t = \sigma(W^{(o)}x_t + U^{(o)}h_{t-1} + b^{(o)}) \tag{3}$$

The candidate value of cell state:

$$u_t = \tanh(W^{(u)}x_t + U^{(u)}h_{t-1} + b^{(u)}) \tag{4}$$

Update cell state:

$$c_t = i_t \odot u_t + f_t \odot c_{t-1} \tag{5}$$

$$h_t = o_t \odot \tanh(c_t) \tag{6}$$

where x_t is the input at the current time step t (such as one word in a sentence), σ denotes the logistic sigmoid function and $\odot$ denotes elementwise multiplication. The superscript of the weight matrices and biase indicate what they are used for. For example, $W^{(i)}$ is a matrix of input. The input gate i_t represents new information we're going to input. The candidate value u_t is created by a tanh layer. i_t multiplied by u_t decides final new information added to the cell state. The forget gate f_t decides which information we're going to throw away from the cell state. f_t multiplied by the last step cell state c_{t-1} filters discarded information. New cell state c_t is obtained by adding these two items. Finally, the cells state c_t is processed through a tanh layer, and then multiplied by the output gate o_t to decide which sections will be output.

Tree-LSTM is a variant of LSTM which combines text structure and LSTM. Tai et al. (2015) has two variants of LSTM. One is Child-Sum Tree -LSTM; the other is N-ary Tree-LSTM. Both two networks can integrate information of multiple child nodes. Each Tree-LSTM node t includes an input gate i_t , an output gate o_t , a cell unit c_t and a hidden output h_t like standard LSTM network. The difference is that the standard LSTM has only one forget gate, but the parent node has a forget gate f_{tk} for each child node k in Tree-LSTM. The difference allows Tree-LSTM to selectively integrate child nodes' information. The input x_t represents the word vector in sentence parse tree. The word vector is equivalent to the head word if Tree-LSTM network is based on dependency tree, while the word vector is the leaf node vector if Tree-LSTM network is based on constituency tree.

Given a tree, the child nodes collection of node t is $C(t)$. The equations of Child-Sum Tree-LSTM are expressed as follows:

$$h_{t-1} = \sum_{k \in C(t)} h_k \tag{7}$$

$$i_t = \sigma(W^{(i)} x_t + U^{(i)} h_{t-1} + b^{(i)}) \tag{8}$$

$$o_t = \sigma(W^{(o)} x_t + U^{(o)} h_{t-1} + b^{(o)}) \tag{9}$$

$$f_{tk} = \sigma(W^{(f)} x_t + U^{(f)} h_k + b^{(f)}) \tag{10}$$

$$u_t = \tanh(W^{(u)} x_t + U^{(u)} h_{t-1} + b^{(u)}) \tag{11}$$

$$c_t = i_t \odot u_t + \sum_{k \in C(t)} f_{tk} \odot c_k \tag{12}$$

$$h_t = o_t \odot \tanh(c_t) \tag{13}$$

In equation 7 and 12, h_k and c_k respectively represent the hidden output and cell state of child node k. Since the last step hidden output of Child-Sum Tree-LSTM depends on the sum of hidden states of child nodes, it's particularly suitable for tree-structure network with disordered child nodes and multi-branch structure. Dependency tree is a good choice, because the number of dependents of head is arbitrary and we don't need to distinguish which is left child or right child. Tree-LSTM based on the dependency tree such as Child-Sum Tree-LSTM is called Dependency Tree-LSTM. Other equations are like equation 1-6. The other network is N-ary Tree-LSTM. In the network, the number of branches of tree is at most N. Child nodes of tree are ordered, which can be labeled from 1 to N. For any node

t , memory cell and hidden output of its k-th child are respectively denoted as h_{tk} and c_{tk}. The equations of N-ary Tree-LSTM are shown as follows:

$$i_t = \sigma(W^{(i)}x_t + \sum_{\varepsilon=1}^{N} U_{\varepsilon}^{(i)} h_{t\varepsilon} + b^{(i)}) \tag{14}$$

$$f_{tk} = \sigma(W^{(f)}x_t + \sum_{\varepsilon=1}^{N} U_{k\varepsilon}^{(f)} h_{t\varepsilon} + b^{(f)}) \tag{15}$$

$$o_t = \sigma(W^{(o)}x_t + \sum_{\varepsilon=1}^{N} U_{\varepsilon}^{(o)} h_{t\varepsilon} + b^{(o)}) \tag{16}$$

$$u_t = \tanh(W^{(u)}x_t + \sum_{\varepsilon=1}^{N} U_{\varepsilon}^{(u)} h_{t\varepsilon} + b^{(u)}) \tag{17}$$

$$c_t = i_t \odot u_t + \sum_{\varepsilon=1}^{N} f_{t\varepsilon} \odot c_{t\varepsilon} \tag{18}$$

$$h_t = o_t \odot \tanh(c_t) \tag{19}$$

In general, the network structure is constructed based on the constituency Tree. The granularity of learning is finer than dependency tree. More detailed information can be found in the original paper Tai et al. (2015). But on binary tree structure leads these two tree structures similar. Model parameters of them are also very similar. When these two networks are simplified to chain structure, the equations 7-13 and 14-19 will transform into standard LSTM equations 1-6.

3.2. The Rhetorical Structure Theory (RST)

Unlike Tree-LSTM, RST can clearly points out the importance of different segments of text. RST was proposed by W. Man and S. Thompson in a paper "Rhetorical Structure Theory: A Theory of Text Organization" Pang and Lee (2005) in 1987. It's a descriptive theory that describes text organizations especially the relations between different parts of text. RST proposes 23 kinds of rhetorical relations, such as attribution, contrast and condition. RST relation splits a piece of text into core segments and peripheral segments, called nuclei and satellites. A sentence's RST relation structure is described in Figure 2. The sentence is "My girlfriend said that she doesnt like watching animated film, but very like this movie." In Figure 2, horizontal line indicates boundaries of each layer of RST roles in hierarchical rhetorical structure. The arrow points from the satellite to the nucleus. Description over horizontal line shows the specific relations between nuclei and satellites, including one or several of 23 kinds of rhetorical relations mentioned above. Emotional words in sentence are marked in red. Contrast relationship is described by line without arrows in both ends, which indicates parallel relations.

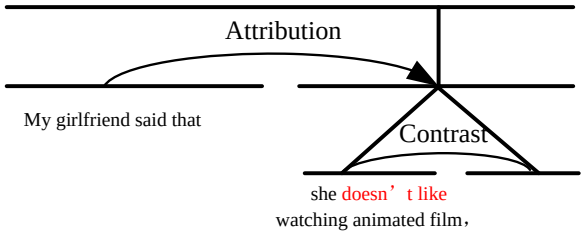


Figure 2: the RST relation structure of a sentence.

3.3. RST-LSTM network

Text structure is one of the key research directions for sentiment analysis tasks. Recent work uses RST for text parsing in order to distinguish important text segments from less important ones in terms of their contribution to the overall sentiment Hogenboom et al. (2015). The RST-LSTM model we propose is described in Figure 3 which includes three parts, RST parse tree, RST-LSTM topology structure and RST-LSTM network. In Figure 3(a), RST parse tree considers the full hierarchical rhetorical structure of sentences, rather than segments' rhetorical roles. We use the two rhetorical structure parsers Surdeanu et al. (2015) proposed. Both parsers use the same underlying framework, but the characteristics of syntax are different. One is based on constituent parse tree (abbreviated as C); the other is based on dependency parse tree (abbreviated as D). The dependency parse tree and the constituent parse tree of the sentence we used are the same. You can parse another sentences by using dependency and constituent RST tools [1]. Compared with other RST discourse analysis tools, these two RST discourse parsers use a simple modular structure. Not only their parsing speed is fast, but also they achieve better results. Whether based on constituent syntax or dependency syntax, RST parse tree describes the master-slave relationship between nodes of each layer: nucleus segments and satellite segments; or the parallel relationship: as shown in Figure 3(a), two segments in the right bottom are of equal importance. We can say they are all nucleus segments. Their relations information is stored in their parent node.

Many previous researches have used RST parse tree, but the combination method for sentences is a simple weighted method, instead of automatic learning. Inspired by Zhu et al. (2015), we use LSTM method to replace simply weighted method. Each rectangle in Figure 3 (b) represents a LSTM unit. We transform part of RST-LSTM topology structure into RST-LSTM network. In Figure 3(c) RST-LSTM network, for a non-leaf node t, we assume that it has two child nodes, denoted as (n, s) which represent two segments of the full sentences.

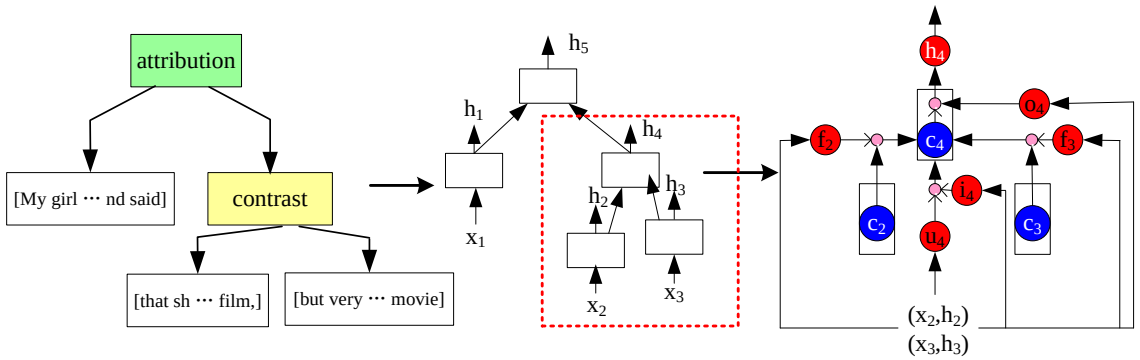
RST-LSTM equations are expressed as follows:

$$i_t = \sigma(W_n^{(i)} x_n + W_s^{(i)} x_s + U_n^{(i)} h_{tn} + U_s^{(i)} h_{ts} + b^{(i)}) \tag{20}$$

$$f_{tn} = \sigma(W_n^{(f)} x_n + W_s^{(f)} x_s + U_{nn}^{(f)} h_{tn} + U_{ns}^{(f)} h_{ts} + b^{(f)}) \tag{21}$$

$$f_{ts} = \sigma(W_n^{(f)} x_n + W_s^{(f)} x_s + U_{sn}^{(f)} h_{tn} + U_{ss}^{(f)} h_{ts} + b^{(f)}) \tag{22}$$

1. http://agathon.sista.arizona.edu:8080/discp/



(a) RST parse tree (b) RST-LSTM topology structure (c) RST-LSTM network

Figure 3: RST-LSTM model.

$$o_t = \sigma(W_n^{(o)} x_n + W_s^{(o)} x_s + U_n^{(o)} h_{tn} + U_s^{(o)} h_{ts} + b^{(o)}) \tag{23}$$

$$u_t = \tanh(W_n^{(u)} x_n + W_s^{(u)} x_s + U_n^{(u)} h_{tn} + U_s^{(u)} h_{ts} + b^{(u)}) \tag{24}$$

$$c_t = i_t \odot u_t + f_{tn} \odot c_{tn} + f_{ts} \odot c_{ts} \tag{25}$$

$$h_t = o_t \odot \tanh(c_t) \tag{26}$$

x_n and x_s respectively represent the two segment vectors which are the input of the parent node. h_{tn} and h_{ts} are the hidden output of nodes (n, s). c_{tn} and c_{ts} are the cell state of nodes (n, s). Equation 20-26 are derived from equation 1-7. Parent node has one forget gate for each node. The key of the model is to selectively integrate information of child nodes to generate parent vector representation. The reason why equation 21 and 22 share some weights in the sigmoid function is they are all the input of the forget gate of the parent node. $w_n^{(f)} x_n + w_s^{(f)} x_s$ is corresponding to x_t in equation 10 and 15. As the equation 25 shows, the cell state is updated by adding three items. The first item represents final new information were going to store in the cell state. The second item represents the nucleus information were going to carry in the cell state. The third item represents the satellite information were going to keep in the cell state. RST-LSTM enhances the nucleus information and weakens or even skips the satellite information to update cell state by their corresponding forget gates. In Figure 3(b), a leaf node represents a segment after RST parsing. We need to denote the segment as a continuous distributed representation. In addition to relations between segments, RST also save syntax structure of discourse analytic dependence. Using the syntax structure, we aim at generating the initial vector representation of segments. Detail will be seen in next section 4.2.

RST-LSTM model not only uses RST parse tree to distinguish the nucleus segment and the satellite segment of sentences instead of only considering sentences as temporal sequences, but also use LSTM structural characteristics to automatically enhance the nucleus information and filter the satellite information of text, instead of simple weighted of different segments' rhetorical roles. The model combines RST and LSTM, which makes both of them realizing their own advantage by the greatest degree.

4. Experiments

4.1. dataset

The dataset we use in the section is English movie review dataset MR Pang and Lee (2005). Because RST analysis tool currently only can handle data in English, so we only use the English dataset classification to validate our model. Stanford Sentiment Treebank (SST) Socher et al. (2013) is sorted from MR dataset by parsing, tagging and other treatments. The SST dataset has 11855 sentences. There are 215,154 independent phrases after parsing and internal nodes of phrases have been labeled. In this section, we use the SST dataset to do two kinds of tasks. One is the fine-grained sentiment classification task. Sentences are classified into five ranks: very negative, negative, neutral, positive, very positive. The other is a common binary classification task. Sentences are classified into positive or negative. Binary classification task needs exclude the neutral sentences, so it only has 9613 sentences in dataset. In fine-grained classification task and binary classification task, training set / validation set / test set are designed as Table 1.

Table 1: Dataset partition.

	Training set	Validation set	Test set	Total number
Fine-grained	8544	1101	2210	11855
Binary	6920	872	1821	9613

4.2. Training details

We use the following methods to generate the initial segment vector:

- Random initialization. We conduct random initialization for each segment $x \in R^N$, and then save it to a vector. The vector is continuously updated during training.
- Using pre-trained word vector trained by Glove Pennington et al. (2014). By simple linear combination of word vectors, the segment vector is gotten.
- Leveraging Tree-LSTM methods for semantic combination from words to segments. Because both RST parser and Tree-LSTM are based on the results of constituent and dependency syntax, we use the following methods: using Tree-LSTM composition method from words to segments to get initial segment vector and using RST-LSTM combination method from segments to sentence to get sentence representation. To make the two methods consistent, If RST parser conduct RST discourse analysis through the result of constituent syntax, we use N-ary Tree-LSTM method to get segment vector. If RST parser conducts RST discourse analysis through the result of dependency syntax, we use Child-Sum Tree-LSTM method to get segment vector.
- Using RST-LSTM method regardless of from words to segments or from segments to the whole sentence. The combination from words to segments uses equations 21-26 and segments' internal structure to calculate according to the results of constituent parsing (binary). x_n and x_s represent the word vector. From segments to sentence also use equations 21-26 to calculate, while x_n and x_s represent different segments.

Among these methods, the first, the second and the forth only use C Parser because of the better parsing result. In the third settings, we use both two parsers and make comparison about their experiment results. We can learn the sentiment distribution of each segment by adding a softmax layer on each node. But we only focus on the sentence-level sentiment analysis. When we have learnt the final hidden output h_t (root node) by RST-LSTM network. The forecast distribution of the entity is as follows:

$$p_\theta(y|\{x\}_t) = softmax(W^{(l)}h_t + b^{(l)}) \tag{27}$$

At the root node t, we use a softmax classifier to predict the label y given the root node input $\{x\}_t$. h_t is the input of softmax layer, $W^{(l)}$ and $b^{(l)}$ are the weight and biase parameters which need to optimize during training. Objective function of the model is the negative log-likelihood value:

$$J(\theta) = -\frac{1}{m}\sum_{t=1}^{m}\log p_\theta(y_t|\{x\}_t) + \frac{\lambda}{2}\|\theta\|^2 \tag{28}$$

the variable m is the number of sentences. The variable λ is the parameter of the L2 regularization designed for avoiding over fitting problem, which is consistent with previous work Tai et al. (2015).

4.3. Experiment results

In order to verify the classification results of the RST-LSTM model, we use the SST Socher et al. (2013) dataset to train the existing sentiment classification systems, such as RAE Socher et al. (2011b), MV-RNN Socher et al. (2012), RNTN Socher et al. (2013), Tree-LSTM Tai et al. (2015). The experimental results of fine-grained classification task and binary classification task are shown in Table 2. In experiments, the initial word vectors are trained by Glove Pennington et al. (2014). The dimension of word embedding is 300. Because the memory of our previous machine is 4G, we can only use a small word vector file, an 822MB word vector file trained on Wikipedia 2014 + Gigaword5, 989.9 MB after decompressing. While the size of the word vector file used in Tree-LSTM original work is 2.03GB, 5.3GB after decompression. In addition, it is possibly because the machine configuration and the operating environment are not the same, there a little difference between our results and the results of original works. In order to compare experiment results under the same conditions, results in this section are all gotten by using our machine.

Our methods (the last two methods) have the best performance. The RST-LSTM (constituent) means the third method to initial segment vector described in Section 4.2. Using Tree-LSTM (constituent) composition method to get initial segment vector and using RST-LSTM combination method to get sentence representation. RST-LSTM means using the forth method in Section 4.2. Using RST-LSTM method regardless of from words to segments or from segments to the whole sentence. The results show that RST-LSTM (constituent) has slightly higher classification accuracy than the RST-LSTM no matter in fine-grained classification task or binary classification task, but RST-LSTM (constituent) training speed is slower than the RST-LSTM, which will be validated next.

We set 25 as the size of mini-batch. Learning rate is set to 0.05 by continuous debugging. The segment vector is updated by learning rate 0.1. The L2 regularization parameter λ is

Table 2: Classification accuracy of different models.

Methods	Fine-grained	Binary
RAE	41.1	81.0
MV-RNN	42.8	81.6
RNTN	44.7	83.9
Tree-LSTM(constituent)	46.8	85.3
RST-LSTM(constituent)	50.3	87.6
RST-LSTM	49.7	86.8

10^{-4} and the number of iterations is set to 10. Each epoch's training time of fine-grained classification task is shown in Figure 4. Using the same settings, each epoch's training time of binary classification task is shown in Figure 5.

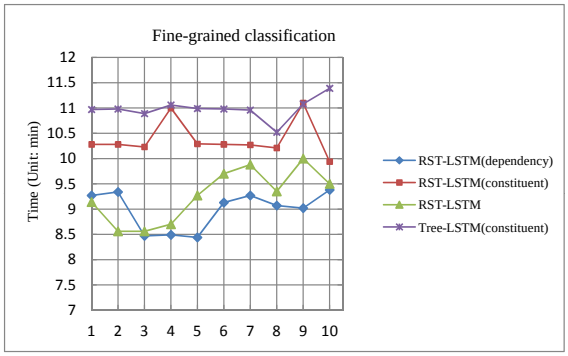


Figure 4: Each epoch's training time of fine-grained classification.

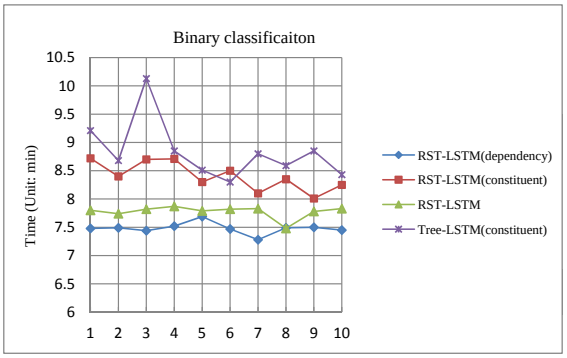


Figure 5: Each epoch's training time of binary classification.

The following relations can be concluded from these two figures, whether it's fine-grained classification task or binary classification task.

$$T(RST-LSTM(dependency)) < T(RST-LSTM) < T(RST-LSTM(constituent)) < T(Tree-LSTM(constituent)) \tag{29}$$

According equation 29, training time of RST-LSTM is between Tree-LSTM (constituent) which adopts N-ary Tree-LSTM to get initial segment vector and Tree-LSTM(dependency) which adopts Child-Sum Tree-LSTM to get initial segment vector. And whether it's RST-LSTM (constituent) or RST-LSTM, the training time of them are shorter than Tree-LSTM (constituent) according to equation 29.

For more accurate comparative experiments, we redo the experiments on machine which has larger memory space. The size of the word vector file is up to 5.3G after decompressing, which is consistent with the original work Tai et al. (2015). By comparing these two different experimental results, we can conclude that the size of word vector file has effect on classification accuracy. The second experiment results can be seen in Table 3.

Table 3: Classification accuracy of different models(2).

Methods	Fine-grained	Binary
RAE	43.2	82.0
MV-RNN	44.3	82.3
RNTN	45.6	85.4
Tree-LSTM(constituent)	48.8	86.9
RST-LSTM(constituent)	51.7	88.8
RST-LSTM	50.8	88.0

The results of Table 3 indicates that the classification accuracy of these models have improved compared with Table 2 when increasing the size of word vector file. But the results of sentiment classification are improved over all models. The performance of our method is still higher than other models.

we also validate our model on another new dataset [2] which is introduced by Dong et al. (2014). we use the standard train/test splits 6248/692. Training set contains negative/neutral/positive three categories with 1561/3127/1560. Test set contains negative/neutral/positive three categories with 172/346/692. We change the dimension of the word vector to 100, training 10 epoches. Other experiment settings are the same as before. The result of our model(using RST-LSTM to get segment vector) have 0.708 classification accuracy, while the RAE model have 0.591 classification accuracy.

4.4. model analysis

In Section 4.2, we propose four methods to get the initial segment vector. The third method contains two settings. We analyze the impact of these methods on classification accuracy

2. http://goo.gl/5Enpu7

of RST-LSTM model. The model uses a fixed 300 dimension of segment vector, updated by learning rate 0.1 in training process, and other parameters are the same as before. The sentiment classification results of different methods have been shown in following Table 4.

Table 4: Classification results of different settings.

Method of initializing segment vector	Fine-grained	Binary
Random initialization	41.1	80.6
Linear combination of Glove word vector	43.3	82.2
Tree-LSTM training (dependency)	48.5	85.7
Tree-LSTM training (constituent)	50.3	87.6
RST-LSTM	49.7	86.8

In Table 4, constituent and dependency in parentheses indicate that the semantic combination from words to segments use N-ary Tree-LSTM method or Child-Sum Tree-LSTM method. The classification accuracy of the last three ways is significantly higher than the first two ways. The classification accuracy of linear combination of Glove word vector is also not much higher than random initialization. The reason may be that the word vector trained by Glove is captured more semantic relations rather than sentiment. and the long segments composed by a relatively large number of words. It's rough that these large segment vectors are calculated only by a simple linear combination. Segment vectors obtained by Tree-LSTM (dependency) aren't as good as the result based on Tree-LSTM (constituent). But Tree-LSTM (dependency) has shorter training time, which is consistent with the original work Tai et al. (2015). The classification result of using RST-LSTM method to get segment vector is between results of these two Tree-LSTM ways and the training time is also between these two ways. That is to say training time of RST-LSTM is longer than Tree-LSTM (dependency), but shorter than Tree-LSTM (constituent).

5. Conclusion and Future Works

5.1. Conclusion

During learning process of text features, further emphasizing the structure of text helps to improve the feature representation, thereby improving the classification accuracy. We propose RST-LSTM model that build deep learning network on RST parse tree. Each node in RST parse tree represents the nucleus segment or the satellite segment. The model can selectively integrate information of child nodes to update cell state by using their corresponding forget gates and captures long-distance relationship. Introducing RST into the sentence-level sentiment classification task and deeply studying the relations features of text make semantic combination based on text structure more accurate. The model is trained on Stanford Sentiment Treebank dataset Socher et al. (2013). Experimental results show that the model not only has higher classification accuracy than the state-of-the-art methods, which improves the accuracy about 2 percent, but also trains quickly.

5.2. Future Works

These methods construct deep learning network based on the parse tree of text, which lead that good or bad feature representations largely depend on syntax tree structure. So the optimization of syntax parse tree is an important exploring direction in the future. In addition, labels have played a crucial role for the classifier training. There are more mature English sentiment dataset, such as the SST Socher et al. (2013) dataset. However, the Chinese sentiment library need further study.

Acknowledgments

This research is supported by the National Nature Science Foundation of China under Grants 61472258, 61402294, National Key Technology Research and Development Program of the Ministry of Science and Technology of China (2014BAH28F05), Technology Planning Project from Guangdong Province (2014B010118005), Science and Technology Foundation of Shenzhen City under Grants JCYJ20140509172609162 and JCYJ20130329102032059.

References

Jos M. Chenlo, Alexander Hogenboom, and David E. Losada. Rhetorical structure theory for polarity estimation: An experimental study. *Data & Knowledge Engineering*, 94: 135–147, 2014.

Kyunghyun Cho, Bart Van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *Computer Science*, 2014. URL http://arxiv.org/abs/1406.1078.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *Meeting of the Association for Computational Linguistics*, pages 49–54, 2014.

Vanessa Wei Feng and Graeme Hirst. Text-level discourse parsing with rich linguistic features. In *Meeting of the Association for Computational Linguistics: Long Papers*, pages 60–68, 2012.

Alex Graves. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

Alexander Hogenboom, Flavius Frasincar, Franciska De Jong, and Uzay Kaymak. Using rhetorical structure in sentiment analysis. *Communications of the Acm*, 58(7):69–77, 2015.

Jing Huang and B. Kingsbury. Audio-visual deep learning for noise robust speech recognition. In *In 2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7596–7599, 2013.

Moshe Koppel and Jonathan Schler. The importance of neutral examples for learning sentiment. *Computational Intelligence*, 22(2):100109, 2006.

Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *International Conference on Machine Learning*, pages 609–616, 2009.

Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *Meeting of the Association for Computational Linguistics*, pages 302–308, 2014.

T. Mikolov, W. T. Yih, and G. Zweig. Linguistic regularities in continuous space word representations. *In HLT-NAACL*, pages 746–751, 2013a.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *Computer Science*, 2013b. URL http://arxiv.org/abs/1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 26:3111–3119, 2013c.

Jeff Mitchell and Mirella Lapata. Composition in distributional models of semantics. *Cognitive Science A Multidisciplinary Journal*, 34(8):1388–429, 2010.

Georgios Paltoglou and Mike Thelwall. A study of information retrieval weighting schemes for sentiment analysis. In *ACL 2010, Proceedings of the Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden*, pages 1386–1395, 2010.

Bo Pang and Lillian Lee. Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales. *Arxiv Cornell University Library*, pages 115–124, 2005.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Acl-02 Conference on Empirical Methods in Natural Language Processing*, pages 79–86, 2002.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, 2014.

R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP*, pages 1631–1642, 2013.

Richard Socher, Chiung Yu Lin, Andrew Y. Ng, Christopher D. Manning, Richard Socher, Chiung Yu Lin, and Andrew Y. Ng. Parsing natural scenes and natural language with recursive neural networks. In *International Conference on Machine Learning, ICML 2011, Bellevue, Washington, Usa, June 28 - July*, pages 129–136, 2011a.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John Mcintyre Conference Centre, Edinburgh, Uk, A Meeting of Sigdat, A Special Interest Group of the ACL*, pages 151–161, 2011b.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. Semantic compositionality through recursive matrix-vector spaces. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211, 2012.

Radu Soricut and Daniel Marcu. Sentence level discourse parsing using syntactic and lexical information. In *Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 149–156, 2003.

Mihai Surdeanu, Thomas Hicks, and Marco A. Valenzuela-Escarcega. Two practical rhetorical structure theory parsers. In *Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies: Software Demonstrations*, 2015. URL http://www.aclweb.org/anthology/N/N15/N15-3001.pdf.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. *Computer Science*, 2015.

Mei Tu, Yu Zhou, and Chengqing Zong. A novel translation framework based on rhetorical structure theory. In *Meeting of the Association for Computational Linguistics*, pages 370–374, 2013.

Lanjun Zhou, Binyang Li, Wei Gao, Zhongyu Wei, and Kam Fai Wong. Unsupervised discovery of discourse relations for eliminating intra-sentence polarity ambiguities. In *Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John Mcintyre Conference Centre, Edinburgh, Uk, A Meeting of Sigdat, A Special Interest Group of the ACL*, pages 162–171, 2011.

Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. Long short-term memory over tree structures. In *ICML*, pages 1604–1612, 2015.