Tree Communication Models for Sentiment Analysis

Yuan Zhang and **Yue Zhang**
School of Engineering, Westlake University, China
Institute of Advanced Technology, Westlake Institute for Advanced Study
chuengyuenn@gmail.com, zhangyue@westlake.edu.cn

Abstract

Tree-LSTMs have been used for tree-based sentiment analysis over Stanford Sentiment Treebank, which allows the sentiment signals over hierarchical phrase structures to be calculated simultaneously. However, traditional tree-LSTMs capture only the bottom-up dependencies between constituents. In this paper, we propose a tree communication model using graph convolutional neural network and graph recurrent neural network, which allows rich information exchange between phrases constituent tree. Experiments show that our model outperforms existing work on bidirectional tree-LSTMs in both accuracy and efficiency, providing more consistent predictions on phrase-level sentiments.

1 Introduction

There has been increasing research interest investigating sentiment classification over hierarchical phrases (Tai et al., 2015; Zhu et al., 2015; Looks et al., 2017; Teng and Zhang, 2017). As shown in Figure 1, the goal is to predict the sentiment class over a sentence and each phrase in its constituent tree. There have been methods that classify each phrase independently (Li et al., 2015; McCann et al., 2017). However, sentiments over hierarchical phrases can have dependencies. For example, in Figure 1, both sentences have a phrase "an awesome day", but the polarities of which are different according to their sentence level contexts.

To better represent such sentiment dependencies, one can encode a constituency tree holistically using a neural encoder. To this end, tree-structured LSTMs have been investigated as a dominant approach (Tai et al., 2015; Zhu et al., 2015; Gan and Gong, 2017; Yu et al., 2017; Liu et al., 2016). Such methods work by encoding hierarchical phrases bottom-up, so that sub constituents can be used as inputs for representing a

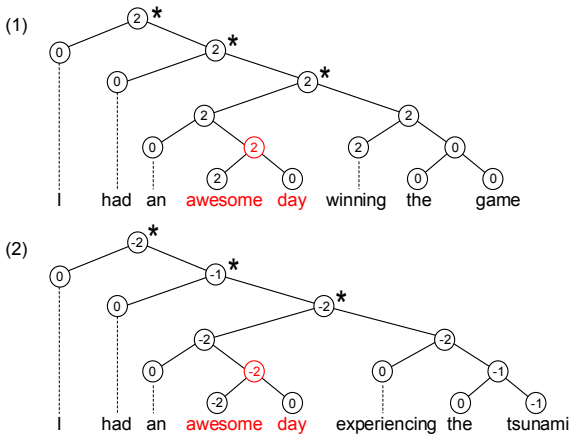


Figure 1: Examples of tree-based sentiment.

constituent. However, they cannot pass information from a constituent node to its children, which can be necessary for cases similar to Figure 1. In this example, sentence level information from top-level nodes is useful for disambiguating "an awesome day". Bi-directional tree LSTMs provide a solution, using a separate top-down LSTM to augment a tree-LSTM (Teng and Zhang, 2017). This method has achieved highly competitive accuracies, at the cost of doubling the runtime.

Intuitively, information exchange between tree nodes can happen beyond bottom-up and top-down directions. For example, direct communication between sibling nodes, such as ("an awesome day", "winning the game") and ("an awesome day", "experiencing the tsunami") can also bring benefits to tree representation. Recent advances of graph neural networks, such as graph convolutional neural network (GCN) (Kipf and Welling, 2016; Marcheggiani and Titov, 2017) and graph recurrent neural network (GRN) (Beck et al., 2018; Zhang et al., 2018b; Song et al., 2018) offer rich node communication patterns over graphs. For relation extraction, for example, GCNs have

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 3518–3527
Florence, Italy, July 28 - August 2, 2019.

been shown superior to tree LSTMs for encoding a dependency tree (Zhang et al., 2018c)

We investigate both GCNs and GRNs as tree communication models for tree sentiment classification. In particular, initialized with a vanilla tree LSTM representation, each node repeatedly exchanges information with its neighbours using graph neural networks. Such multi-pass information exchange can allow each node to be more informed about its sentence-level context through rich communication patterns. In addition, the number of time steps does not scale with the height of the tree. To allow better interaction, we further propose a novel time-wise attention mechanism over GRN, which summarizes the representation after each communication step.

Experiments on Stanford Sentiment Treebank (SST; Socher et al. 2013) show that our model outperforms standard bottom-up tree-LSTM (Zhu et al., 2015; Looks et al., 2017) and also recent work on bidirectional tree-LSTM (Teng and Zhang, 2017). In addition, our model allows a more holistic prediction of phase-level sentiments over the tree with a high degree of node sentiment consistency. To our knowledge, we are the first to investigate graph NNs for tree sentiment classification, and the first to discuss phrase level sentiment consistency over a constituent tree for SST. We release our code and models at https://github.com/fred2008/TCMSA.

2 Related Work

Bi-directional Tree-LSTM Paulus et al. (2014) capture bidirectional information over a binary tree by propagating global belief down from the tree root to leaf nodes. Miwa and Bansal (2016) adopt a bidirectional dependency tree-LSTM model by introducing a top-down LSTM path. Teng and Zhang (2017) propose a first bidirectional tree-LSTM for constituent structures, by building a top-down tree-LSTM with estimations of head lexicons. Compared with their work, we achieve information interaction using an asymptotically more efficient algorithm, which performs node communication simultaneously across a whole tree.

Graph Neural Network Scarselli et al. (2009) propose graph neural network (GNN) for encoding an arbitrary graph structure. Kipf and Welling (2016) use graph convolutional network to learn node representation for graph structure. Marcheggiani and Titov (2017) and Bastings et al. (2017) extend the use of graph convolutional network (GCN) to NLP tasks. In particular, they use GCN to learn dependency-syntactic word representation for semantic role labeling and machine translation, respectively. Zhang et al. (2018b) use a graph recurrent network (GRN) to model sentences. Beck et al. (2018) and Song et al. (2018) use a graph recurrent network for learning representation of abstract meaning representation (AMR) graphs. Our work is similar in utilizing graph neural network for NLP. Compared with their work, we apply GNN to constituent trees. In addition, we propose a novel time-wise attention mechanism on GRN to combine recurrent time steps dynamically.

3 Baseline

We take standard bottom-up tree-LSTMs as our baseline. Tree-LSTM extends sequence-LSTM by utilizing 2 previous states for modeling a left child node and a right child node, respectively, in a recurrent state transition process. Formally, a tree-LSTM calculates a cell state through an input gate, an output gate and two forget gates at each time step. In particular, at time step t, the input gate i_t and the output gate o_t are calculated respectively as follows:

$$\begin{aligned} i_t = \sigma\big(&W^L_{hi}h^L_{t-1} + W^R_{hi}h^R_{t-1} \\ &+ W^L_{ci}c^L_{t-1} + W^R_{ci}c^R_{t-1} + b_i\big), \\ o_t = \sigma\big(&W^L_{ho}h^L_{t-1} + W^R_{ho}h^R_{t-1} \\ &+ W^L_{co}c^L_{t-1} + W^R_{co}c^R_{t-1} + b_o\big), \end{aligned}$$

where W^L_{hi}, W^R_{hi}, W^L_{ci}, W^R_{ci}, b_i, W^L_{ho}, W^R_{ho}, W^L_{co}, W^R_{co} and b_o are parameters of the input gate and the output gate, respectively.

The forget gates of the left node f^L_t and the right node f^R_t are calculated respectively as:

$$\begin{aligned} f^L_t = \sigma\big(&W^L_{hf_L}h^L_{t-1} + W^R_{hf_L}h^R_{t-1} \\ &+ W^L_{cf_L}c^L_{t-1} + W^R_{cf_L}c^R_{t-1} + b_{f_L}\big), \\ f^R_t = \sigma\big(&W^L_{hf_R}h^L_{t-1} + W^R_{hf_R}h^R_{t-1} \\ &+ W^L_{cf_R}c^L_{t-1} + W^R_{cf_R}c^R_{t-1} + b_{f_R}\big), \end{aligned}$$

where $W^L_{hf_L}$, $W^R_{hf_L}$, $W^L_{cf_L}$, $W^R_{cf_L}$, b_{f_L}, $W^L_{hf_R}$, $W^R_{hf_R}$, $W^L_{cf_R}$, $W^R_{cf_R}$ and b_{f_R} are parameters.

The cell candidate $\tilde{C}_t$ is dependent on both c^L_{t-1} and c^R_{t-1}:

$$\tilde{C}_t = \tanh\left(W^L_{hC}h^L_{t-1} + W^R_{hC}h^R_{t-1} + b_C\right)$$

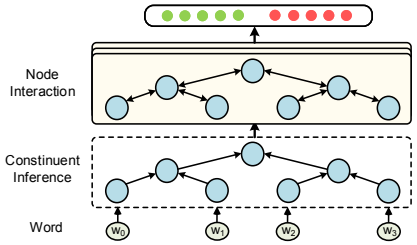


Figure 2: Tree communication model.

where W_{hC}, W_{hC}^R and b_C are model parameters.

Based on the two previous cell states c_{t-1}^L and c_{t-1}^R, the cell state of the current node c_t is calculated as:

$$c_t = f_t^l \otimes c_{t-1}^L + f_t^R \otimes c_{t-1}^R + i_t \otimes \tilde{C}_t,$$

where f_t^l is the forget gate of the left child node, f_t^R is the forget gate of the right child node, $\tilde{C}_t$ is the cell candidate.

Finally, the hidden state h_t of the current node is calculated based on the current cell c_t and the output gate o_t:

$$h_t = o_t \otimes \tanh(c_t)$$

Limitation Tree-LSTM models capture only bottom-up node dependencies. Specifically, for a node j, the hidden representation h_j^{tree} is dependent on the descendant nodes only. Formally,

$$h_j^{tree} = f(h_{d_{j0}}, h_{d_{j1}}, \cdots, h_{d_{jk}}),$$

where d_j is the set of descendant nodes of node j.

Bi-directional Solution A bidirectional tree-LSTM (Bi-tree-LSTM) takes a bottom-up tree-LSTM as a first step, performing a top-down tree communication process. Teng and Zhang (2017) is one example.

4 Tree Communication Models

Our tree communication models (TCM) take a trained tree LSTM as an initial state, performing information exchange using graph neural network (GNN). Thus h_j is dependent on all related neighborhood nodes rather than only descendant nodes:

$$h_j = f(h_{r_{j0}}, h_{r_{j1}}, \cdots, h_{r_{jk}}),$$

where r_j is the set of all relevant nodes of node j. Such node can be the full tree with sufficient communication.

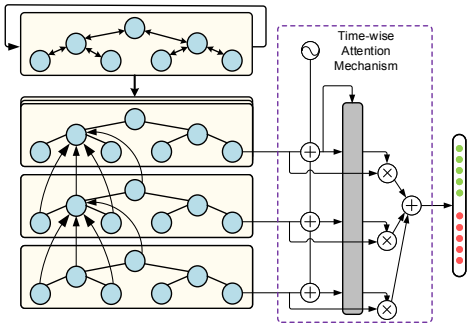


Figure 3: Recurrent tree communication model.

In particular, given a constituent tree, for each constituent node j, the initial state h'_j is obtained using a tree-LSTM:

$$h'_j = \textit{treeLSTM}(h'_{\textit{left}(j)}, c'_{\textit{left}(j)}, h'_{\textit{right}(j)}, c'_{\textit{right}(j)}),$$

where h'_j is the hidden state of the node j, c'_j is the cell state of node j, $\textit{left}(j)$ denote the left child of node j, $\textit{right}(j)$ denotes the right child of node j.

As shown in Figure 2, a TCM performs information exchange between a constituent node j with its neighbor nodes in three channels:

- A **self-to-self channel** transfers information from node j to itself. The input for the channel is represented as $x_j^{self} = h'_j$, where h'_j is the initial state of tree communication model.
- A **bottom-up channel** transfers information from lower level nodes to upper-level nodes. The inputs for the channel are represented as $x_j^{left} = h'_{\textit{left}(j)}$, $x_j^{right} = h'_{\textit{right}(j)}$, where $\textit{left}(j)$ and $\textit{right}(j)$ denote the left child and the right child of node j, respectively. x_j^{up} is the sum of inputs from bottom up: $x_j^{up} = x_j^{left} + x_j^{right}$.
- A **top-down channel** transfers information from parent nodes to child nodes. The input for the channel is represented as: $x_j^{down} = h'_{\textit{prt}(j)}$, where $\textit{prt}(j)$ denotes the parent node of node j.

When tree communications are executed repeatedly, each node receives information from an increasingly larger context. We explore a convolutional tree communication model (CTCM) and a recurrent tree communication model (RTCM), which are based on GCN (Marcheggiani and Titov, 2017) and GRN (Song et al., 2018), respectively. Both models allow node interactions in a tree to be performed in parallel, and thus are

computationally efficient. The time complexity to achieve additional interaction of TCMs are O(1), in contrast to O(n) by top-down tree-LSTM.

4.1 Convolutional Tree Communication Model

We apply the strategy of Marcheggiani and Titov (2017), where multiple convolutional layers can be used for information combination. In particular, for the k-th layer, transformed inputs are obtained by linear transformation for each channel:

$$x_j^{k,self} = W^{k,self} h_j^{k-1,self} + b^{k,self},$$
$$x_j^{k,d} = W^{k,up} h_j^{k-1,d} + b^{k,up}, d \in \{left, right\},$$
$$x_j^{k,down} = W^{k,down} h_j^{k-1,down} + b^{k,down},$$

where $W^{k,e}$ and b^e ($e \in \{self, up, down\}$) are model parameters, and $h_j^{k-1,e}$ ($e \in \{self, left, right, down\}$) is the hidden state of last layer for node $e(j)$. The initial $h_j^{-1,e}$ are the inputs of three channels x_j^e defined earlier.

Following Marcheggiani and Titov (2017), for each edge type $e \in \{self, left, right, down\}$, we apply edge-wise gate to all inputs:

$$g_j^{k,e} = \sigma(W_g^{k,e} h_j^{k,e} + b_g^{k,e}),$$

where $W_g^{k,e}$ and $b_g^{k,e}$ are model parameters.

The final representation of node j is:

$$h_j^k = f(\sum_e x_j^{k,e} \otimes g_j^{k,e}).$$

4.2 Recurrent Tree Communication Model

We take the stategy of Song et al. (2018). The structure of RTCM shows in Figure 3. For each recurrent step t, the hidden states from the last recurrent step are taken to calculate the cell state of the current state. In particular, for node j, the hidden state of the previous step can be divided into the last hidden state h_j^{self} from self-to-self channel, the last hidden state $h_{t-1,j}^{up}$ from bottom-up channel and the last hidden state $h_{t-1,j}^{down}$ from the top-down channel:

$$h_{t-1,j}^{self} = h_{t-1,j},$$
$$h_{t-1,j}^{up} = h_{t-1,left(j)} + h_{t-1,right(j)},$$
$$h_{t-1,j}^{down} = h_{t-1,prt(j)}.$$

We calculate gate and state values based on the inputs and last hidden states from the three information channels. The input gate i_t^j and the forget gate f_t^j are defined as:

$$i_j^t = \sigma\Big(W_i^{self} x_j^{self} + W_i^{up} x_j^{up} + W_i^{down} x_j^{down} + U_i^{self} h_{t-1,j}^{self} + U_i^{up} h_{t-1,j}^{up} + U_i^{down} h_{t-1,j}^{down} + b_i\Big),$$
$$f_j^t = \sigma\Big(W_f^{self} x_j^{self} + W_f^{up} x_j^{up} + W_f^{down} x_j^{down} + U_f^{self} h_{t-1,j}^{self} + U_f^{up} h_{t-1,j}^{up} + U_f^{down} h_{t-1,j}^{down} + b_f\Big),$$

where W_i^{self}, W_i^{up}, W_i^{down}, U_i^{self}, U_i^{up}, U_i^{down}, b_i, W_f^{self}, W_f^{up}, W_f^{down}, U_f^{self}, U_f^{up}, U_f^{down} and b_f are parameters of input and forget gate.

The cell candidate $\tilde{C}_t^j$ is defined as:

$$\tilde{C}_j^t = \sigma\Big(W_C^{self} x_j^{self} + W_C^{up} x_j^{up} + W_C^{down} x_j^{down} + U_C^{self} h_{t-1,j}^{self} + U_C^{up} h_{t-1,j}^{up} + U_C^{down} h_{t-1,j}^{down} + b_C\Big),$$

where W_C^{self}, W_C^{up}, W_C^{down}, U_C^{self}, U_C^{up}, U_C^{down} and b_C are parameters of cell candidate.

The current cell state is calculated as:

$$c_j^t = f_j^t \otimes c_j^{t-1} + i_j^t \otimes \tilde{C}_j^t,$$

The output gate o_t^j is defined as:

$$o_j^t = \sigma\Big(W_o^{self} x_j^{self} + W_o^{up} x_j^{up} + W_o^{down} x_j^{down} + U_o^{self} h_{t-1,j}^{self} + U_o^{up} h_{t-1,j}^{up} + U_o^{down} h_{t-1,j}^{down} + b_o\Big),$$

where W_o^{self}, W_o^{up}, W_o^{down}, U_o^{self}, U_o^{up}, U_o^{down} and b_o are model parameters.

The final hidden h_j^t is calculated through the current cell state c_j^t and the output gate o_j^t:

$$h_j^t = o_j^t \otimes \tanh(c_j^t).$$

4.2.1 Time-wise attention

Both GRN and GCN calculate a sequence of incrementally more abstract representations $c_j^1, c_j^2, ...c_j^t$ for each node c_j. We further introduce a novel attention scheme to GRN. Intuitively, each recurrent step in RTCM learns a different level of abstraction. For a constituent node higher in the tree or on the leaf, more recurrent steps may be needed to learn the interaction between nodes. Accordingly, we use an adaptive recurrence mechanism to learn a dynamic node representation through attention structure (Bahdanau et al., 2014).

Our method first encodes a recurrent-step-sensitive hidden state with positional embedding:

$$h_t^{j,depth} = h_t^j + e_t^p,$$

where $h_t^{j,depth}$ is the recurrent-step-sensitive hidden state for node j on t-th step, e_p is positional encoding of the recurrence steps.

Inspired by Vaswani et al. (2017), a static representation is used for the positional encoding $e_p(t)$, which does not require training:

$$e^p_{t,2k} = \sin(t/10000^{2k/d_{emb}}),$$
$$e^p_{t,2k+1} = \cos(t/10000^{2k/d_{emb}}),$$

t is the index of recurrent steps, $e_{t,m}$ is the m-th dimension of positional embedding, and d_{emb} is the dimension of embedding.

We learn the weight w_t for the t-th recurrent step by the relationship between $h_T^{j,depth}$ and $h_t^{j,depth}$:

$$w'_{j,t} = h_T^{j,depth} \cdot h_t^{j,depth},$$
$$w_{j,t} = \frac{\exp(w'_{j,i})}{\sum_{t=0}^{T-1} \exp(w'_{j,t})}.$$

The final state can be represented as a weighted sum of the hidden states obtained after different recurrent steps:

$$h^j = \sum_{t=0}^{T-1} w_t h_t^j.$$

5 Decoding and Training

Following Looks et al. (2017) and Teng and Zhang (2017), we perform softmax classification on each node according to the last hidden state:

$$o = \text{softmax}(Mh + b)$$

where M and b are model parameters. For training, negative log-likelihood loss is computed over each o locally, and accumulated over the tree.

6 Experiments

We test the effectiveness of TCM by comparing its performance with a standard tree-LSTM (Zhu et al., 2015) as well as a state-of-the-art bidirectional tree-LSTM (Teng and Zhang, 2017). A series of analysis is conducted for measuring the holistic representation of sentiment in a tree via phrase-level sentiments consistency.

6.1 Data

We use the Stanford Sentiment Treebank (SST; Socher et al. 2013), which is a dataset of movie reviews originally from Pang and Lee (2005) annotated at both the clause level and the sentence level. Following Zhu et al. (2015) and Teng and Zhang (2017), we perform both fine-grained sentiment classification and binary classification. For the former, the dataset was annotated for 5 levels of sentiment: strong negative, negative, neutral, positive, and strong positive. For the latter, the data was labeled with positive sentiment and negative sentiment. We adopt a standard dataset split following Tai et al. (2015); Teng and Zhang (2017). Table 1 lists the data statistics.

Corpus	SST-5	SST-2
Classes	5	2
Sentences	11,855	9,613
Phrases	442,629	137,988
Tokens	227,242	185,621

Table 1: Data statistics.

6.2 Experimental Settings

Hyper-parameters We initialize word embeddings using GloVe (Pennington et al., 2014) 300-dimensional embeddings. Embeddings are fine-tuned during training. The size of LSTM hidden states are set to 300. We thus fix the number to 9.

Training In order to obtain a good representation for an initial constituent state, we first train an independent bottom-up tree-LSTM, over which we train our tree communication models. To avoid over-fitting, we adopt dropout on the embedding layer, with a rate of 0.5. Training is done on mini-batches through Adagrad (Duchi et al., 2011) with a learning rate of 0.05. We adopt gradient clipping with a threshold of 1.0. The L2 regularization parameter is set to 0.001.

6.3 Development Experiments

Hyper-parameters We investigate the effect of recurrent steps of RTCM as shown in Block A of Table 2. As the number of steps increases from 1, the accuracy increases, showing the effectiveness of tree node communication. A recurrent step of 9 gives the best accuracies, and a larger number of steps does not give further improvements. This is consistent with observations of Song et al. (2018), which shows that sufficient context information can be collected over a small number of iterations.

The effectiveness of TCM Block B in Table 2

Block	Model	SST-5	SST-2
A	3 Step RTCM	83.1	92.3
	6 Step RTCM	83.2	92.7
	9 Step RTCM	83.4	92.9
	18 Step RTCM	83.2	92.8
B	Tree-LSTM	82.9	92.4
	CTCM	83.3	92.8
	RTCM	83.4	92.9
	RTCM+attention	83.5	93.3

Table 2: Phrase level performances on the dev set.

shows the performance of different models. Tree-LSTMs with different TCMs outperform the baseline tree-LSTM on both datasets. In addition, the time-wise attention mechanism in Section 4.2.1 improves performance on both SST-5 and SST-2. In the remaining experiments, we use RTCM with time wise-attention.

6.4 Final Results

Table 3 shows the overall performances for sentiment classification on both SST-5 and SST-2. We report accuracies on both the sentence level and the phrase level. Compared with previous methods based on constituent tree-LSTM, our model improves the preformance on different datasets and settings. In particular, it outperforms BiConLSTM (Teng and Zhang, 2017), which use bidirectional tree-LSTM. This demostrates the advantage of graph neural networks compared to a top-down LSTM for tree communication. Our model gives the state-of-the-art accuracies on phrase-level settings. Note that we do not leverage character representation or external resources such as sentiment lexicons and large-scale corpuses.

There has also been work using large-scale external datasets to improve performance. McCann et al. (2017) pretrain their model on large parallel bilingual datasets and exploit character n-gram features. They report an accuracy of 53.7 on sentence-level SST-5 and an accuracy of 90.3 on sentence-level SST-2, which are lower than our model. Peters et al. (2018) pretrain a language model with character convolutions on a large-scale corpus and report an accuracy of 54.7 on sentence-level SST-5, which is slightly higher than our model. Large-scale pretraining is orthogonal to our method. For a fair comparison, we do not list their results on Table 3.

We further analyze the performance with respect to different sentence lengths. Figure 4 shows the results. On both datasets, the performance of tree-LSTM on sentences of lengths less than 10 ($l = 5$ in the figure) is much better than that of longer sentences. There is a tendency of decreasing accuracies as the sentence length increases. As the length of sentences increases, there are longer-range dependencies along the depth of tree structure, which is more difficult to model than short sentences.

It can be seen that the improvement of TCM over tree-LSTM model is larger with increasing sentence length. This shows that longer sentences can benefit more from rich tree communication.

Model	SST-5		SST-2	
	R	P	R	P
RNTN (S13)	45.7	80.7	85.4	87.6
BiLSTM (L15)	49.8	83.3	86.7	-
ConTree (LZ15)	49.9	-	88.0	-
ConTree (Z15)	50.1	-	-	-
ConTree (L15)	50.4	83.4	86.7	-
ConTree (T15)	51.0	-	88.0	-
Disan (S18)	51.7	-	-	-
RL LD/HS-LSTM (Z18)	50.0	-	87.8	-
NTI-SLSTM (MY17)	53.1	-	89.3	-
ConTree(Fold) (L17)	52.3	-	89.4	-
BiConTree (TZ17)	53.5	83.5	**90.3**	92.8
RTC + attention	**54.3**	**83.6**	**90.3**	**93.4**

Table 3: Final results (R-Root, P-Phrase). S13 – Socher et al. (2013); L15 – Li et al. (2015); LZ15 – Le and Zuidema (2015); Z15 – Zhu et al. (2015); T15 – Tai et al. (2015); S18 – Shen et al. (2018); Z18 – Zhang et al. (2018a); MY17 – Munkhdalai and Yu (2017); L17 – Looks et al. (2017); TZ17 – Teng and Zhang (2017)

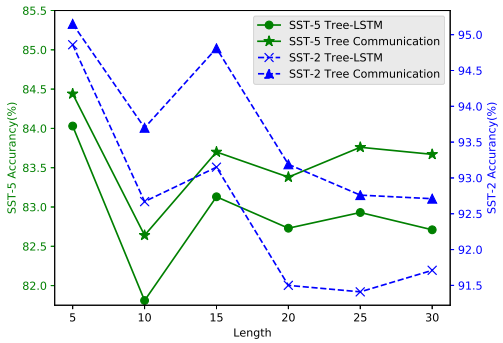
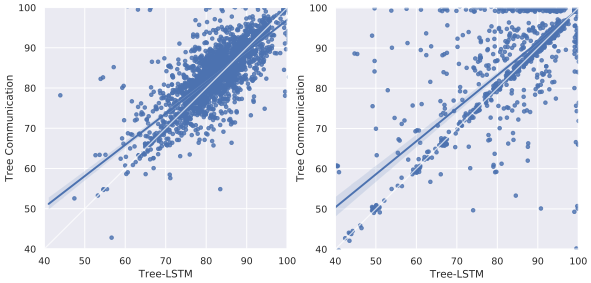


Figure 4: Sentiment classification accuracies against the sentence length. The accuracy for each length l is calculated on the test set sentences length in the bin $[l, l + 5]$.



(a) SST-5 (b) SST-2

Figure 5: Sentence-level phrase accuracy (*SPAcc*) scatter plot. Each dot represents a sentence in the test dataset. Its x-coordinate and y-coordinate are *SPAcc* for predicted phrase label sequence of the baseline model and TCM respectively. The blue line is a linear regression line of all dots.

Dataset	α	Baseline	Our Model	Diff.
SST-5	1.0	3.5	3.7	+0.2
	0.9	18.9	21.2	+2.3
	0.8	67.6	71.4	+3.8
SST-2	1.0	56.0	61.4	+5.4
	0.9	18.9	21.2	+4.6
	0.8	67.6	71.4	+2.0

Table 4: Rates of holistically-labeled sentences with sentence-level phrase accuracy *SPAcc* $\geqslant \alpha$.

6.5 Disscusion

Sentence-level performance To further compare performances of holistic phrase sentiment classification on the sentence level, we measure the accuracy on the sentence level. We define sentence-level phrase accuracy (*SPAcc*) of a sentence as: $\textit{SPAcc} = n_{correct}/n_{total}$, where n_{total} is the total number of phrases in the sentence, and $n_{correct}$ is the number of correct sentiment predictions in the sentence. For each sentence of test dataset, taking *SPAcc* of the corresponding label sequence resulting from the baseline model as the x-coordinate and *SPAcc* of the corresponding label sequence resulting from TCM as the y-coordinate, we draw a scatter plot with a regression line as shown in Figure 5. The regression line is inclined towards the top-left, indicating that TCM can improve the performance on holistic phrase classifications over a whole sentence.

If the *SPAcc* of a sentence is high, the sentence is more holistically-labeled. Table 4 shows the statistics on the rate of holistically-labeled sentences with *SPAcc* $\geqslant \alpha$ (*SPAcc*-α). The rate of holistically-labeled sentences for TCM is higher

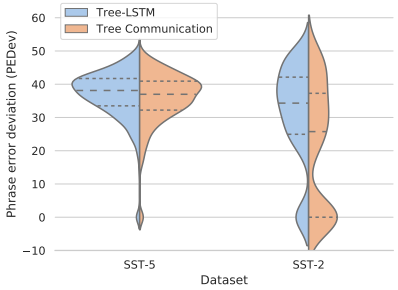


Figure 6: Deviation of node errors for each tree.

Dataset	Metric	Baseline	TCM	Diff.
SST-5	mean	36.9	35.7	-1.2
	median	38.1	37.0	-1.1
SST-2	mean	31.4	21.8	-9.6
	median	34.3	25.8	-8.3

Table 5: Deviation statistics. Values in units of $\times 10^{-2}$

than that for tree-LSTM on both SST-5 and SST-2 for different values of α. It demonstrates that TCM labels the constituent nodes of the whole tree better than the tree-LSTM model, thanks to more information exchange between phrases in a tree.

Consistency between nodes To compare the sentiment classification consistency of phrases in each sentence, we define a metric, phrase error deviation (*PEDev*), to measure the deviation among the error of labels for one sentence:

$$\textit{PEDev}(\hat{y}, y) = \sqrt{\frac{1}{N}\sum_{i=0}^{N-1}\left[d(\hat{y}_i, y_i) - \bar{d}\,\right]^2},$$

where $d(\hat{y}_i, y_i)$ is the Hamming distance between the i-th predicted label and the i-th ground truth label. $\bar{d}$ is the mean value of $d(\hat{y}_i, y_i)$. Since $d(\hat{y}_i, y_i) \in [0, 1]$, $\textit{PEDev}(\hat{y}, y) \in [0, 0.5]$.

For an input sentence, if all the predicted labels are the same as the ground truth, or all the predicted labels are different from the ground truth, $\textit{PEDev}(\hat{y}, y) = 0$, which means that the sentence is labeled with the maximum consistency. On the contrary, if the predicted labels of some phrases are the same as ground truth while others are not, $\textit{PEDev}(\hat{y}, y)$ is high. Table 5 lists the statistics on $\textit{PEDev}(\hat{y}, y)$ of the baseline model and our model for all the test sentences on SST-5 and SST-2. The mean and median of $\textit{PEDev}(\hat{y}, y)$ of TCM are much less than those of the baseline tree-LSTM model. In addition, as Figure 6 shows, compared with the $\textit{PEDev}(\hat{y}, y)$ distribution of the tree-LSTM model, the distribution of TCM is relatively less in value. It demonstrates that TCM

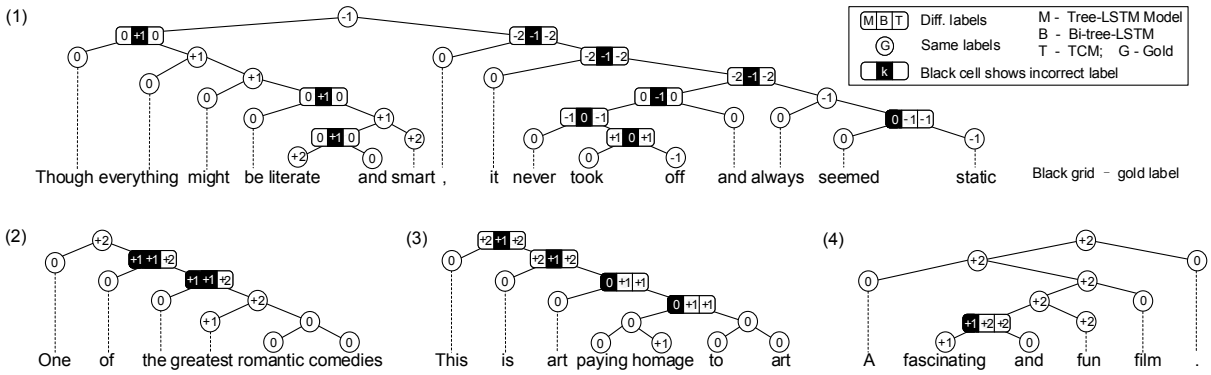


Figure 7: Sentiment classification samples.

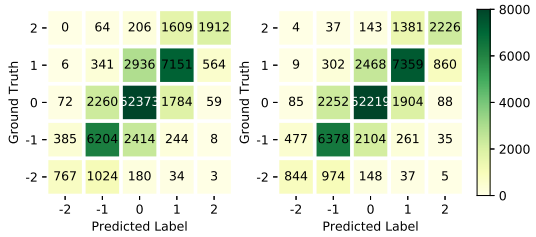
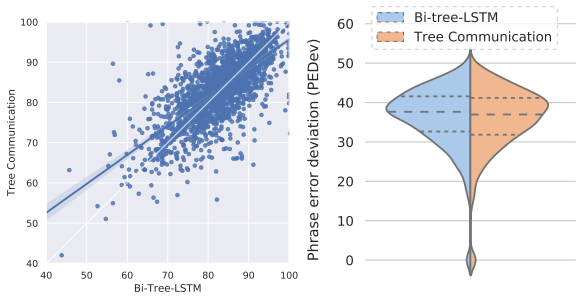


Figure 8: Confusion matrix on SST-5 phrase-level test dataset for tree-LSTM (left) and TCM (right).

Metrics	BTL	TCM	Diff.
SPAcc, $\alpha = 1.0$	3.2	3.7	+0.5
SPAcc, $\alpha = 0.9$	20.0	21.2	+1.2
SPAcc, $\alpha = 0.8$	70.7	71.4	+0.7
PEDev-mean	36.4	35.7	-0.7
PEDev-median	37.6	37.0	-0.6

Table 6: Sentence-level phrase accuracy (*SPAcc*) and phrase error deviation (*PEDev*) comparison on SST-5 between bi-tree-LSTM and TCM.



(a) (b)

Figure 9: Sentence-level phrase accuracy (a) and deviation of node errors (b) comparison on SST-5 between bi-tree-LSTM and TCM.

improves the consistency of phrase classification for each sentence.

Confusion matrix Figure 8 shows the confusion matrix on the SST-5 phrase-level test set for tree-LSTM (left) and TCM (right). Compared with tree-LSTM, the accuracies of most sentiment labels by TCM increase (the accuracy of the neutral label slightly decreases by 0.3%), indicating that TCM is strong in differentiating fine-grained sentiments in global and local contexts.

6.6 Comparison with Bi-tree-LSTM

Table 6 shows the sentence-level phrase accuracy (*SPAcc*) and phrase error deviation (*PEDev*) comparison on SST-5 between bi-tree-LSTM and TCM, respectively. TCM outperforms bi-tree-LSTM on all the metrics, which demonstrates that TCM gives more consistent predictions of sentiments over different phrases in a tree, compared to top-down communication. This shows the benefit of rich node communication.

Figure 9 shows a scatter chart and a deviation chart comparision between the two models, in the same format as Figure 5 and Figure 6, respectively. As shown in Figure 9a, the errors of TCM and bitree-LSTM are scattered, which shows that different communication patterns influence sentiment prediction. The final observation is consistent with Table 6.

6.7 Case Study

Figure 7 shows four samples on SST-5. In the first sentence, the phrase "seemed static" itself bares the neutral sentiment. However, it has a negative sentiment in the context. The tree-LSTM model captures the sentiment of the phrase bottom-up, therefore giving the neutral sentiment. In con-

trast, TCM considers larger contexts by repeated node interaction. The phrase "seemed static" receives information from the constituents "never took off" and "Though everything might be literate and smart" through their common ancestor nodes, leading to the correct result. Although bi-tree-LSTM predicts these sentiments of the phrase "seemed static" and the whole sentence correctly, it gives more incorrect results on the phrase level. The other sentences in Figure 7 show similar trends. From the samples we can find that TCM provides more consistent predictions on phrase-level sentiments thanks to its better understanding of different contexts.

7 Conclusion

We investigated two tree communication models for sentiment analysis, leveraging recent advances in graph neural networks for information exchange between nodes in a baseline tree-LSTM model. Both GCNs and GRNs are explored and compared, with GRNs showing better accuracies. We additionally propose a novel time-wise attention mechanism to further improve GRNs. Results on standard benchmark show that graph NNs give better results compared to bi-directional tree LSTMs, providing more consistent predictions over phrases in one sentence. To our knowledge, we are the first to leverage graph neural network structures for enhancing tree-LSTMs, and the first to discuss tree-level sentiment consistency using a set of novel metrics.

8 Acknowledgments

The corresponding author is Yue Zhang. We thank the anonymous reviewers for their valuable comments and suggestions. We thank Zhiyang Teng and Linfeng Song for their work and discussion. This work is supported by a grant from Rxhui Inc[1].

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *EMNLP*.

Daniel Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-sequence learning using gated graph neural networks. In *ACL*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*.

Ling Gan and Houyu Gong. 2017. Text sentiment analysis based on fusion of structural information and serialization information. In *IJCNLP*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Phong Le and Willem Zuidema. 2015. Compositional distributional semantics with long short term memory. In **SEM*.

Jiwei Li, Minh-Thang Luong, Dan Jurafsky, and Eudard Hovy. 2015. When are tree structures necessary for deep learning of representations? In *EMNLP*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Deep multi-task learning with shared memory. In *EMNLP*.

Moshe Looks, Marcello Herreshoff, DeLesley Hutchins, and Peter Norvig. 2017. Deep learning with dynamic computation graphs. In *ICLR*.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *EMNLP*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *NIPS*.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *ACL*.

Tsendsuren Munkhdalai and Hong Yu. 2017. Neural tree indexers for text understanding. In *ACL*.

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

Romain Paulus, Richard Socher, and Christopher D Manning. 2014. Global belief recursive neural networks. In *NIPS*.

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

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL*.

[1]https://rxhui.com

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The graph neural network model. *IEEE Transactions on Neural Networks*.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018. Disan: Directional self-attention network for rnn/cnn-free language understanding. In *AAAI*.

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

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. A graph-to-sequence model for amr-to-text generation. In *ACL*.

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

Zhiyang Teng and Yue Zhang. 2017. Head-lexicalized bidirectional tree lstms. *TACL*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Liang-Chih Yu, Jin Wang, K Robert Lai, and Xuejie Zhang. 2017. Refining word embeddings for sentiment analysis. In *EMNLP*.

Tianyang Zhang, Minlie Huang, and Li Zhao. 2018a. Learning structured representation for text classification via reinforcement learning. In *AAAI*.

Yue Zhang, Qi Liu, and Linfeng Song. 2018b. Sentence-state lstm for text representation. In *ACL*.

Yuhao Zhang, Peng Qi, and Christopher D Manning. 2018c. Graph convolution over pruned dependency trees improves relation extraction. *arXiv preprint arXiv:1809.10185*.

Xiaodan Zhu, Parinaz Sobihani, and Hongyu Guo. 2015. Long short-term memory over recursive structures. In *ICML*.