Long Short-Term Memory Over Recursive Structures

Xiaodan Zhu XIAODAN.ZHU@NRC-CNRC.GC.CA
National Research Council Canada, 1200 Montreal Road M-50, Ottawa, ON K1A 0R6 CANADA

Parinaz Sobhani PSOBH090@UOTTAWA.CA
School of Electrical Engineering and Computer Science, University of Ottawa, 800 King Edward Avenue, Ottawa, ON K1N 6N5 CANADA

Hongyu Guo HONGYU.GUO@NRC-CNRC.GC.CA
National Research Council Canada, 1200 Montreal Road M-50, Ottawa, ON K1A 0R6 CANADA

Abstract

The chain-structured long short-term memory (LSTM) has showed to be effective in a wide range of problems such as speech recognition and machine translation. In this paper, we propose to extend it to tree structures, in which a memory cell can reflect the history memories of multiple *child cells* or multiple *descendant cells* in a recursive process. We call the model S-LSTM, which provides a principled way of considering long-distance interaction over hierarchies, e.g., language or image parse structures. We leverage the models for semantic composition to understand the meaning of text, a fundamental problem in natural language understanding, and show that it outperforms a state-of-the-art recursive model by replacing its composition layers with the S-LSTM memory blocks. We also show that utilizing the given structures is helpful in achieving a performance better than that without considering the structures.

1. Introduction

Recent years have seen a revival of the long short-term memory (LSTM) (Hochreiter & Schmidhuber, 1997), with its effectiveness being demonstrated on a wide range of problems such as speech recognition (Graves et al., 2013), machine translation (Sutskever et al., 2014; Cho et al., 2014), and image-to-text conversion (Vinyals et al., 2014), among many others, in which history is summarized and coded in the *memory blocks* in a full-order fashion.

Recursion is a fundamental process associated with many problems—a recursive process and the structure it forms are common in different modalities. For example, semantics of sentences in human languages is arguably to be carried not merely by a linear concatenation of words; instead, sentences often have structures (Manning & Schütze, 1999). Image understanding, as another example, may benefit from recursive modeling over structures, which yielded the state-of-the-art performance on tasks like scene segmentation (Socher et al., 2011).

In this paper, we extend LSTM to tree structures, in which we attempt to learn memory blocks that can reflect the history memories of multiple *child cells* and hence multiple *descendant cells*. We call the model S-LSTM. Compared with previous recursive neural networks (Socher et al., 2013; 2012), S-LSTM has the potentials of avoiding *gradient vanishing* and hence may model long-distance interactions over trees. This is a desirable characteristic as many of such structures are deep. S-LSTM can be viewed as considering together a recursive neural network and a recurrent neural network[1]. In brief, S-LSTM wires memory blocks in a partial-order tree structures instead of in a full-order sequence as in a chain-structured LSTM.

We leverage the S-LSTM model to solve a semantic composition problem that learns the meaning for a piece of texts—learning a good representation for the meaning of a span of text is core to automatically understanding human languages. More specifically, we experiment with the mod-

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

[1]As both of them can be shortened to be RNN, in the rest of this paper we refer to a Recurrent Neural Network as RNN and a Recursive Neural Network as RvNN.

els on the Stanford Sentiment Treebank dataset (Socher et al., 2013) to determine the sentiment for different granularities of phrases in a tree. The dataset has favorable properties: in addition to being a benchmark for much previous work, it provides with human annotations at all nodes of the trees, enabling us to comprehensively explore the properties of S-LSTM. We experimentally show that S-LSTM outperforms a state-of-the-art recursive model by simply replacing the original tensor-enhanced composition with the S-LSTM memory block that we propose here. We show that utilizing the given structures is helpful in achieving a better performance than that without considering the structures.

2. Related Work

Recursive neural networks Recursion is a fundamental process in different modalities. In recent years, recursive neural networks (RvNN) have been introduced and demonstrated to achieve the state-of-the-art performances on different problems such as semantic analysis in natural language processing and image segmentation (Socher et al., 2013; 2011). These networks are defined over recursive tree structures—a tree node is a vector computed from its children. In a recursive fashion, the information from the leaf nodes of a tree and its internal nodes are combined in a bottom-up manner through the tree. Derivatives of errors are computed with backpropagation over structures (Goller & Kuchler, 1996).

In addition, the literature has also included many other efforts of applying feedforward-based neural network over structures, including (Goller & Kuchler, 1996; Chater, 1992; Starzyk & He, 2007; Hammer et al., 2004; Guo et al., 2014), amongst others. For instance, Legrand and Collobert leveraged neural networks over greedy syntactic parsing (Pinheiro & Collobert, 2014). In (Irsoy & Cardie, 2014), a deep recursive neural network was proposed, and in (Hermann & Blunsom, 2013) composition was performed with the consideration of linguistic motivations specifically from the combinatory categorial grammar (CCG) formalism. Nevertheless, over the often deep structures, the networks are potentially subject to the vanishing gradient problem, resulting in difficulties in leveraging long-distance dependencies in the structures. In this paper, we propose the S-LSTM model that wires memory blocks in recursive structures. We compare our model with the RvNN models presented in (Socher et al., 2013), as we directly replaced the tensor-enhanced composition layer at each tree node with a S-LSTM memory block. We show the advantages of our proposed model in achieving significantly better results.

Recurrent neural networks and LSTM Unlike a feedforward network, a recurrent neural network (RNN) shares their hidden states across time. The sequential history is summarized in a hidden vector. RNN also suffers from the decaying of gradient, or less frequently, blowing-up of gradient problem. LSTM replaces the hidden vector of a recurrent neural network with *memory blocks* which are equipped with gates; it can in principle keep long-term memory by training proper gating weights (refer to (Graves, 2008) for intuitive illustrations and good discussions), and it has practically showed to be very useful, achieving the state of the art on a range of problems including speech recognition (Graves et al., 2013), digit handwriting recognition (Liwicki et al., 2007; Graves, 2008), and achieve interesting results on statistical machine translation (Sutskever et al., 2014; Cho et al., 2014) and music composition (Eck & Schmidhuber, 2002b;a). In (Graves et al., 2013), a deep LSTM network achieved the state-of-the-art results on the TIMIT phoneme recognition benchmark. In (Sutskever et al., 2014; Cho et al., 2014), a pair of LSTM networks are trained to encode and decode human language for automatic machine translation, which is in particular effective for the more challenging long sentence translation. In (Liwicki et al., 2007; Graves, 2008), LSTM networks are found to be very useful for digit writing recognition because of the network's capability of memorizing context information in a long sequence. In (Eck & Schmidhuber, 2002b;a), LSTM networks are trained to effectively capture global structures of the temporal data. With the memory cells, LSTM is able to keep track of temporally distant events that indicate global music structures. As a result, LSTM can be successfully trained to compose music, where other RNNs have failed to do so.

Although promising results have been observed by applying chain-structured LSTM, many other interesting problems are inherently associated with input structures that are more complicated than a sequence. For example, sentences in human languages are believed to be carried by not merely a linear sequence of words; instead, meaning is thought to interweave with structures. While a sequential application of LSTM may capture structural information implicitly, in practice it sometimes lacks the claimed power. For example, even simply reversing the input sequences may result in significant differences in modeling performances, in tasks such as machine translation and speech recognition. Unlike in previous work, we propose here to directly wire memory blocks in recursive structures. We show the proposed S-LSTM model does utilize the structures and achieve results better than those ignoring such priori structures.

In addition, in parallel to our efforts, independent research (Tai et al., 2015; Le & Zuidema, 2015) has been conducted to explore ideas close to ours. We refer the readers to these coming interesting works as well.

3. The Model

Model brief In this paper, we extend LSTM to structures, in which a memory cell can reflect the history memories of multiple child cells and hence multiple descendant cells in a hierarchical structure. As intuitively showed in Figure 1, the root of the tree can in principle consider information from long-distance interactions over the tree—in this figure, the gray and light-blue leaf. In the figure, the small circle ("$\circ$") or short line ("$-$") at each arrowhead indicates *pass* and *block* of information, respectively. Note that the figure shows a binary case, while in real models a soft version of gating is applied, where a gating signal is in the range of [0, 1], often enforced with a logistic sigmoid function. Through learning the gating signals, as detailed later in this section, S-LSTM provides a principled way of considering long-distance interplays over the input structures.

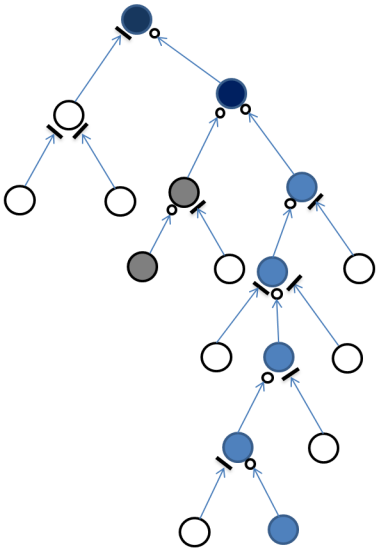

Figure 1. An example of S-LSTM, a long-short term memory network on tree structures. A tree node can consider information from multiple descendants. Information of the other nodes in white are blocked. The small circle ("$\circ$") or short line ("$-$") at each arrowhead indicates a *pass* or *block* of information, respectively, while in the real model the gating is a soft version of gating.

The memory block Each node in Figure 1 is composed of a S-LSTM *memory block*. We present a specific wiring of such a block in Figure 2. Each memory block contains one input gate and one output gate. The number of forget gates depends on the structure, i.e., the number of children of a node. In this paper, we assume there are two children at each nodes, same as in (Socher et al., 2013) and therefore we use their data in our experiments. That is, we have two forget gates. Extending the model to handle a *n-ary* tree is rather straightforward. In addition, it may also deserve to note that on the other hand, binarizing a tree to a binary tree is also a common practice in some domains (e.g., in language parsing) to render structures.

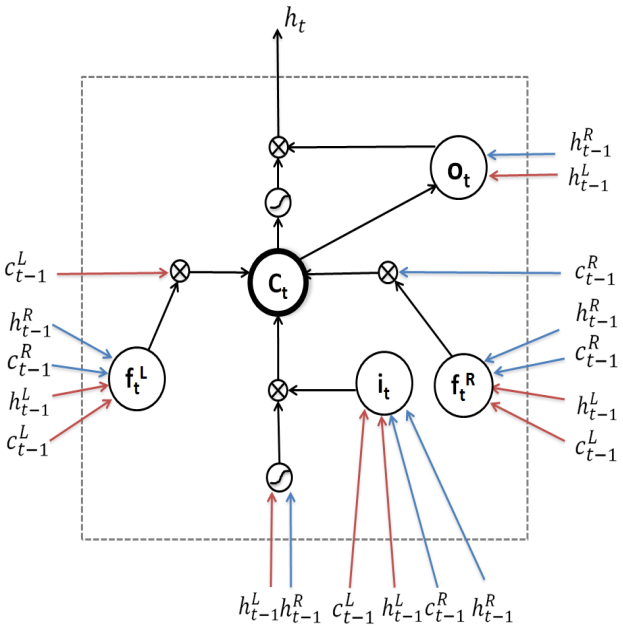

Figure 2. A S-LSTM memory block, consisting of an input gate, two forget gates, and an output gate. Hidden vectors h^*_{t-1} and cell vectors c^*_{t-1} from the left (red arrows) and right (blue arrows) children are deployed to compute c_t and h_t. $\otimes$ denotes a Hadamard product, and the "s" shaped sign is a squashing function (in this paper the *tanh* function).

As shown in the figure, the hidden vectors of the two children, denoted as h^L_{t-1} for the left child and h^R_{t-1} for the right, are taken in as input of the current block. The input gate i_t consider four resources of information: the hidden vectors (h^L_{t-1} and h^R_{t-1}) and cell vectors (c^L_{t-1} and c^R_{t-1}) of its two children. These four sources of information are also used to form the gating signals for the left forget gate f^L_{t-1} and right forget gate f^R_{t-1}, where the weights used to combining them are specific to each of these gates, denoted as different W in the formulas below. Different from the process in a regular LSTM, the cell here considers the copies from both children's cell vectors (c^L_{t-1}, c^R_{t-1}), gated with separated forget gates. The left and right forget gates can be controlled independently, allowing the pass-through of information from children's cell vectors. The output gate o_t considers the hidden vectors from the children and the current cell vector. In turn, the hidden vector h_t and the cell vector c_t of the current block are passed to the parent and are used depending on if the current block is a left or right child of its parent. In this way, the memory cell, through merging the gated cell vectors of the children, can reflect multiple direct or indirect descendant cells. As a result, the long-distance interplays over the structures can be

captured. More specifically, the forward computation of a S-LSTM memory block is specified in the following equations.

$$i_t = \sigma(W^L_{hi}h^L_{t-1} + W^R_{hi}h^R_{t-1} + W^L_{ci}c^L_{t-1} + W^R_{ci}c^R_{t-1} + b_i) \quad (1)$$

$$f^L_t = \sigma(W^L_{hf_l}h^L_{t-1} + W^R_{hf_l}h^R_{t-1} + W^L_{cf_l}c^L_{t-1} + W^R_{cf_l}c^R_{t-1} + b_{f_l}) \quad (2)$$

$$f^R_t = \sigma(W^L_{hf_r}h^L_{t-1} + W^R_{hf_r}h^R_{t-1} + W^L_{cf_r}c^L_{t-1} + W^R_{cf_r}c^R_{t-1} + b_{f_r}) \quad (3)$$

$$x_t = W^L_{hx}h^L_{t-1} + W^R_{hx}h^R_{t-1} + b_x \quad (4)$$

$$c_t = f^L_t \otimes c^L_{t-1} + f^R_t \otimes c^R_{t-1} + i_t \otimes tanh(x_t) \quad (5)$$

$$o_t = \sigma(W^L_{ho}h^L_{t-1} + W^R_{ho}h^R_{t-1} + W_{co}c_t + b_o) \quad (6)$$

$$h_t = o_t \otimes \tanh(c_t) \quad (7)$$

where σ is the element-wise logistic function used to confine the gating signals to be in the range of [0, 1]; f^L and f^R are the left and right forget gate, respectively; b is bias and W is network weight matrices; the sign $\otimes$ is a Hadamard product, i.e., element-wise product. The subscripts of the weight matrices indicate what they are used for. For example, W_{ho} is a matrix mapping a hidden vector to an output gate.

Backpropagation over structures During training, the gradient of the objective function with respect to each parameter can be calculated efficiently via backpropagation over structures (Goller & Kuchler, 1996; Socher et al., 2013). The major difference from that of (Socher et al., 2013) is we use LSTM-like backpropagation, where unlike a regular LSTM, pass of error needs to discriminate between the left and right children, or in a topology with more than two children, needs to discriminate between children. Obtaining the backpropagation formulas is tedious but we list them below to facilitate duplication of our work. We will discuss the specific objective function later in experiments. For each memory block, assume that the error passed to the hidden vector is ϵ^h_t. The derivatives of the output gate δ^o_t, left forget gate $\delta^{f_l}_t$, right forget gate $\delta^{f_r}_t$, and input gate δ^i_t are computed as:

$$\epsilon^h_t = \frac{\partial O}{\partial h_t} \quad (8)$$

$$\delta^o_t = \epsilon^h_t \otimes \tanh(c_t) \otimes \sigma'(o_t) \quad (9)$$

$$\delta^{f_l}_t = \epsilon^c_t \otimes c^L_{t-1} \otimes \sigma'(f^L_t) \quad (10)$$

$$\delta^{f_r}_t = \epsilon^c_t \otimes c^R_{t-1} \otimes \sigma'(f^R_t) \quad (11)$$

$$\delta^i_t = \epsilon^c_t \otimes \tanh(x_t) \otimes \sigma'(i_t) \quad (12)$$

where $\sigma'(x)$ is the element-wise derivative of the logistic function over vector x. Since it can be computed with the activation of x, we abuse the notation a bit to write it over the activated vectors in these equations. ϵ^c_t is the derivative over the cell vector. So if the current node is the left child of its parent, we use Equation (13) to calculate ϵ^c_t, otherwise Formula (14) is used:

$$\epsilon^c_t = \epsilon^h_t \otimes o_t \otimes g'(c_t) + \epsilon^c_{t+1} \otimes f^L_{t+1} + (W^L_{ci})^T\delta^i_{t+1} + (W^L_{cf_l})^T\delta^{f_l}_{t+1} + (W^L_{cf_r})^T\delta^{f_r}_{t+1} + (W_{co})^T\delta^o_t \quad (13)$$

$$\epsilon^c_t = \epsilon^h_t \otimes o_t \otimes g'(c_t) + \epsilon^c_{t+1} \otimes f^R_{t+1} + (W^R_{ci})^T\delta^i_{t+1} + (W^R_{cf_l})^T\delta^{f_l}_{t+1} + (W^R_{cf_r})^T\delta^{f_r}_{t+1} + (W_{co})^T\delta^o_t \quad (14)$$

where $g'(x)$ is the element-wise derivative of the *tanh* function. It can also be directly calculated from the *tanh* activation of x. The superscript T over the weight matrices means matrix transpose.

With derivatives at each gate computed, the derivatives of the weight matrices used in Formula (1)-(7) can be calculated accordingly, which is omitted here. We checked the gradient to ensure the correctness of the implementation of S-LSTM.

Objective over trees The objective function defined over structures can be complicated, which could consider the output structures depending on the properties of problem. Following (Socher et al., 2013), the overall objective function we used to learn S-LSTM in this paper is simply minimizing the overall cross-entropy errors and a sum of that at all nodes.

4. Experiment Set-up

As discussed earlier, recursion is a basic process inherent to many problems. In this paper, we leverage the proposed

model to solve semantic composition for the meanings of pieces of text, a fundamental problem in understanding human languages.

We specifically attempt to determine the sentiment of different granularities of phrases in a tree, within the Stanford Sentiment Treebank benchmark data (Socher et al., 2013). In obtaining the sentiment of a long piece of text, early work often factorized the problem to consider smaller pieces of component words or phrases with bag-of-words or bag-of-phrases models (Pang & Lee, 2008; Liu & Zhang, 2012). More recent work has started to model composition (Moilanen & Pulman, 2007; Choi & Cardie, 2008; Socher et al., 2012; 2013; Kalchbrenner et al., 2014), a more principled approach to modeling the formation of semantics. In this paper, we put the proposed LSTM memory blocks at tree nodes—we replaced the tensor-enhanced composition layer at each tree node presented in (Socher et al., 2013) with a S-LSTM memory block. We used the same dataset, the Stanford Sentiment Treebank, to evaluate the performances of the models. In addition to being a benchmark for much previous work, the data provide with human annotations at all nodes of the trees, facilitating a more comprehensive exploration of the properties of S-LSTM.

4.1. Data Set

The Stanford Sentiment Treebank (Socher et al., 2013) contains about 11,800 sentences from the movie reviews that were originally discussed in (Pang & Lee, 2005). The sentences were parsed with the Stanford parser (Klein & Manning, 2003). Phrases at all the tree nodes were manually annotated with sentiment values. We use the same split of the training and test data as in (Socher et al., 2013) to predict the sentiment categories of the roots (sentences) and all phrases (including sentences). For the root sentiment, the training, development, and test sentences are 8544, 1101, and 2210, respectively. The phrase sentiment task includes 318582, 41447, and 82600 phrases for the three sets. Following (Socher et al., 2013), we also use the classification accuracy to measure the performances.

4.2. Training Details

As mentioned before, we follow (Socher et al., 2013) to minimize the cross-entropy error for all nodes or for roots only, depending on specific experiment settings. For all phrases, the error is calculated as a regularized sum:

$$E(\theta) = \sum_i \sum_j t_j^i \log y^{sen_i}{}_j + \lambda \left\| \theta \right\|_2^2 \qquad (15)$$

where $y^{sen_i} \in \mathbb{R}^{c \times 1}$ is predicted distribution and $t^i \in \mathbb{R}^{c \times 1}$ the target distribution. c is the number of classes or categories, and $j \in c$ denotes the j-th element of the multinomial target distribution; i iterates over nodes, θ are model parameters, and λ is a regularization parameter. We tuned our model against the development data set as split in (Socher et al., 2013).

5. Results

To understand the modeling advantages of S-LSTM over the structures, we conducted four sets of experiments.

Default setting In the default setting, we conducted experiments as in (Socher et al., 2013). Table 1 shows the accuracies of different models on the test set of the Stanford Sentiment Treebank. We present the results on 5-category sentiment prediction at both the sentence level (i.e., the *ROOTS* column in the table) and for all phrases including roots (the *PHRASES* column) [2]. In Table 1, *NB* and *SVM* are naive Bayes and support vector machine classifiers, respectively; *RvNN* corresponds to RNN in (Socher et al., 2013). As described earlier, we refer to recursive neural networks to as RvNN to avoid confusion with recurrent neural networks. RNTN is different from RvNN in that when merging two nodes to obtain the hidden vector of their parent, tensor is used to obtain the second-degree polynomial interactions.

Table 1. Performances (accuracies) of different models on the test set of Stanford Sentiment Treebank, at the sentence level (roots) and the phrase level. † shows the performance are statistically significantly better ($p < 0.05$) than the corresponding models.

MODELS	ROOTS	PHRASES
NB	41.0	67.2
SVM	40.7	64.3
RvNN	43.2	79.0
RNTN	45.7	80.7
S-LSTM	**48.9**†	**81.9**†

Table 1 showed that S-LSTM achieved the best predictive performance, when compared to all the models reported in (Socher et al., 2013). The S-LSTM results reported here were obtained by setting the size of the hidden units to be 100, batch size to be 10, and learning rate to be 0.1. In our experiments, we only tuned these hyper-parameters, and we feel that more finer tuning, such as discriminating the classification weights between the leaves (word embedding) and other nodes, using different numbers of hidden units for the memory blocks (e.g., for the hidden layers of

[2]The Stanford CoreNLP package (http://nlp.stanford.edu/sentiment/code.html) only gives approximate accuracies for 2-category sentiment, which are not included here in the table.

words), or different initializations of word embedding, may further improve the performances reported here.

As discussed earlier in this paper, the model we proposed can be viewed as considering together a recursive and recurrent neural network. Table 1 serves to compare S-LSTM with a directly comparable state-of-the-art recursive model by replacing its composition layers with the S-LSTM memory blocks (later we will also compare S-LSTM with the recurrent version without considering recursive structures.)

Table 2 compares S-LSTM with more state-of-the-art models that reported root-level results. The interesting work described in (Kalchbrenner et al., 2014) uses convolutional networks instead of recursive nor recurrent networks to model sentences. Without using any additionally learned word embedding, the accuracy of our S-LSTM is 48.9%, marginally higher than 48.5% reported in (Kalchbrenner et al., 2014). We also initialized our word representations using publicly available 300-dimensional Glove vectors (Pennington et al., 2013), with which S-LSTM attained an accuracy of 50.1% on the root level task, comparable to 49.8% reported in (Irsoy & Cardie, 2014); the latter proposed a deep recursive network that stacks multiple recursive layers. Note that S-LSTM may be considered orthogonally with such models. While we report here the results using external embedding to initialize words, comparison without that may avoid being potentially confounded by the choices of external embedding and the additional tuning involved.

Table 2. Comparison with more models on five-category classification accuracies on the test set of Stanford Sentiment Treebank at the sentence level (roots).

MODELS	ROOTS
KALCHBRENNER ET AL., 2014	48.5
S-LSTM	48.9
IRSOY & CARDIE, 2014	49.8
S-LSTM (GLOVE-300)	50.1

To evaluate the S-SLTM model's convergence behavior, Figure 3 depicts the converging time during training. More specifically, we show two sub-figures: one for roots (upper sub-figure) and the other for all phrases (lower sub-figure). The curves were obtained by using different initial learning rates corresponding to each best model. From these figures, we can observe that S-LSTM converges faster than the RNTN. For instance, for the phrase-level task, S-LSTM started to converge after about 20 minutes but the RNTN needed over 180 minutes. S-LSTM has much less parameters than RNTN and the forward and backward propagation can be computed efficiently. We also observed that converging time was not very sensitive to the initial learning rates under AdaGrad used in our experiments.

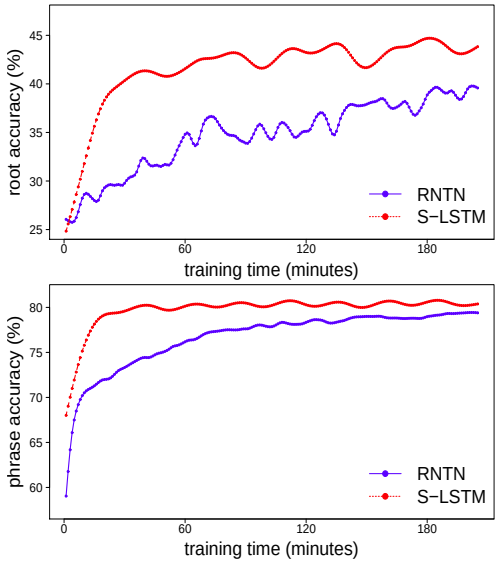

Figure 3. Converging time during training for roots (the upper figure) and for all nodes (the lower figure).

More real-life settings We further compare S-LSTM with RNTN in two more experimental settings. In the first setting we only keep the training signals at the roots to train S-LSTM and RNTN, depicted as model (1) and (2) in Table 3. *ROOT LBLS* besides the model names stands for *root labels*; that is, only the gold labels of the sentence level are used to train the model. In most sentiment analysis circumstances, phrase level annotations are not available: most nodes in a tree are fragments that may not be that interesting; e.g., the fragment *"of a good movie"* [3]. Also, annotating all phrases is expensive. However, these should not be regarded as comments on the value of the Sentiment Treebank. Detailed annotations in the treebank enable much interesting work to be possible, e.g., the study of the effect of negation in changing sentiment (Zhu et al., 2014a).

The second setting, corresponding to model (3) and (4) in Table 3, is only slightly different, in which we keep annotation for the tree leafs as well, to simulate that a sentiment lexicon is available and it covers all leafs (words) ($LEAFLBLS$ beside the model names stands for *leaf labels*), and so there is no out-of-vocabulary concern. Using real sentiment lexicons is expected to have a performance

[3]Phrase-level sentiment analysis is often defined over a very small subset of phrases of interest, such as in the phrase and aspect level tasks defined and studied in (Wilson et al., 2005; Mohammad et al., 2013; Zhu et al., 2014b; Kiritchenko et al., 2014).

between the two settings here.

Results in the table show that in both settings, S-LSTM outperforms RNTN by a large margin. When only root labels are used to train the models, S-LSTM obtains an accuracy of 43.5, compared with 29.1 of RNTN. When the leaf labels are also used, S-LSTM achieves an accuracy of 44.1 and RNTN 34.9. All these improvements are statistically significant ($p < 0.05$). For the RNTN, without supervising signals from the internal nodes, the composition parameters may not be learned well, potentially because the tensor has much more parameters to learn. On the other hand, through controlling its gates, the S-LSTM shows a very good ability to learn from the trees.

Table 3. Performances of models trained with only root labels (the first two rows) and models that use both root and leaf labels (the last two rows).

MODELS	ROOTS
(1) RNTN (ROOT LBLS)	29.1
(2) S-LSTM (ROOT LBLS)	43.5†
(3) RNTN (ROOT + LEAF LBLS)	34.9
(4) S-LSTM (ROOT + LEAF LBLS)	44.1†

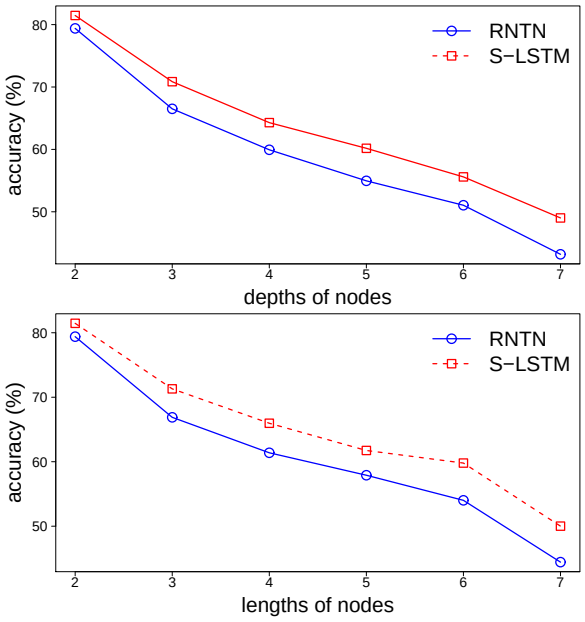

Figure 4. Accuracies at different depths (the upper figure) in the trees, or by different lengths of the phrases (the lower figure).

Performance over different levels of trees In Figure 4, we further depict the performances of models on different levels of nodes in the trees. In this figure, the x-axis corresponds to different depths or lengths and y-axis is accuracy. The *depth* here is defined as the longest distance between the root of a phrase and their descendant leafs. The *Length* is simply the number of words of a node, where *depth* is not necessarily to be *length*—e.g., a balanced tree with 4 leafs has different depths than the unbalanced tree with the same number of leafs. The trends of the two figure are similar. In both figures, S-LSTM performs better at all depths, showing its advantages on nodes at depth. As the deeper levels of the tree tend to have more complicated syntax and semantics, S-LSTM can model such more complicated syntax and semantics better.

Explicit structures vs. no structures Some research efforts in the literature attempt to learn distributed representation by utilizing input structures when available, and others prefer to assume chain-structured recurrent neural networks can actually capture the structures implicitly though a linear coding process. In this paper, we attempt to give some empirical evidences in our experiment setting by comparing several different models. First, a special case for the S-LSTM model is considered, in which no sentential structures are given. Instead, words are read from left to right and combined in that order. We call it left recursive S-LSTM, or S-LSTM-LR in short. Similarly, we also experimented with a right recursive S-LSTM, S-LSTM-RR, in which words are read from right to left instead. Since for these models, phrase-level training signals are not available—the nodes here do not correspond to that in the original Standford Sentiment Treebank, but the roots and leafs annotations are still the same, so we run two versions of our experiments: one uses only training signals from roots and the other includes also leaf annotations.

It can be observed from Table 4 that the given parsing structure helps improve the predictive accuracy. In the case of using only root labels, the left recursive S-LSTM and right recursive S-LSTM have similar performance (40.2 and 40.3, respectively), both inferior to S-LSTM (43.5). When using gold leaf labels, the gaps are smaller, but still, using the parse structure is better. Note that in real applications, where there is out-of-vocabulary issue (i.e., some leafs are not seen in the sentiment dictionaries), the difference between S-LSTM and the recursive version without using the structures are expected to be between the gaps we observed here.

Table 4. Performances of models that do not use the given sentence structures. S-LSTM-LR is a degenerated version of S-LSTM that reads input words from left to right, and S-LSTM-RR reads words from right to left.

MODELS	ROOTS
S-LSTM-LR (ROOT LBLS)	40.2
S-LSTM-RR (ROOT LBLS)	40.3
S-LSTM (ROOT LBLS)	43.5†
S-LSTM-LR (ROOT + LEAF LBLS)	43.1
S-LSTM-RR (ROOT + LEAF LBLS)	43.2
S-LSTM (ROOT + LEAF LBLS)	44.1†

6. Conclusions

We aim to extend the conventional chain-structured long short-term memory to explicitly consider structures. In this paper we particularly study tree structures, in which the proposed S-LSTM memory cell can reflect the history memories of multiple descendants through gated copying of memory vectors. The model provides a principled way to consider long-distance interplays over the structures. We leveraged the model to learn distributed sentiment representations for texts, and showed that it outperforms a state-of-the-art recursive model by replacing its tensor-enhanced composition layers with the S-LSTM memory blocks. We showed that the structure information is useful in helping S-LSTM achieve the state-of-the-art performance.

The research community seems to contain two lines of wisdom; one attempts to learn distributed representation by utilizing structures when available, and the other prefers to believe neural networks can actually capture the structures implicitly though a linear coding process. In this paper, we also attempt to give some empirical evidences toward answering the question. It is at least for the settings of our experiments that the explicit input structures are helpful in inferring the high-level (e.g., root) semantics.

References

Chater, N; Conkey, p. finding linguistic structure with recurrent neural networks. lawrence erlbaum assoc publ. In *Proceedings of the Annual Conference of the Cognitive Science Scoiety. (pp. 402 - 407*, 1992.

Cho, Kyunghyun, van Merrienboer, Bart, Gülçehre, Çaglar, Bougares, Fethi, Schwenk, Holger, and Bengio, Yoshua. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

Choi, Yejin and Cardie, Claire. Learning with compositional semantics as structural inference for subsentential sentiment analysis. In *Proceedings of EMNLP*, pp. 793–801, Honolulu, Hawaii, 2008.

Eck, Douglas and Schmidhuber, Juergen. Finding temporal structure in music: Blues improvisation with lstm recurrent networks. In *Proceedings of IEEE Workshop on Neural Networks for Signal Processing*, pp. 747–756. IEEE, 2002a.

Eck, Douglas and Schmidhuber, Juergen. Learning the long-term structure of the blues. In *Proceedings of ICANN*, pp. 284–289, 2002b.

Goller, Christoph and Kuchler, Andreas. Learning task-dependent distributed representations by backpropagation through structure. In *Proceedings of ICNN*, pp. 347–352, Bochum, Germany, 1996.

Graves, Alex. *Supervised sequence labelling with recurrent neural networks*. PhD thesis, Technische Universitat Munchen, 2008.

Graves, Alex, Mohamed, Abdel-rahman, and Hinton, Geoffrey E. Speech recognition with deep recurrent neural networks. *CoRR*, abs/1303.5778, 2013.

Guo, Hongyu, Zhu, Xiaodan, and Min, Renqiang. A Deep Learning Model for Structured Outputs with High-order Interaction. In *Proceedings of NIPS Workshop on Representation and Learning Methods for Complex Outputs*, 2014.

Hammer, Barbara, Micheli, Alessio, Sperduti, Alessandro, and Strickert, Marc. A general framework for unsupervised processing of structured data. *Neurocomputing*, 57, 2004.

Hermann, Karl Moritz and Blunsom, Phil. The Role of Syntax in Vector Space Models of Compositional Semantics. In *Proceedings of ACL*, pp. 894–904, Sofia, Bulgaria, August 2013.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Computation*, (8):1735–1780, 1997.

Irsoy, Ozan and Cardie, Claire. Deep recursive neural networks for compositionality in language. In *Proceedings of NIPS*, pp. 2096–2104. 2014.

Kalchbrenner, Nal, Grefenstette, Edward, and Blunsom, Phil. A convolutional neural network for modelling sentences. *Proceedings of ACL*, June 2014.

Kiritchenko, Svetlana, Zhu, Xiaodan, and Mohammad, Saif. NRC-Canada-2014: Detecting Aspects and Sentiment in Customer Reviews. In *Proceedings of the International Workshop on Semantic Evaluation*, 2014.

Klein, Dan and Manning, Christopher D. Accurate unlexicalized parsing. In *Proceedings of ACL*, pp. 423–430, Sapporo, Japan, 2003.

Le, Phong and Zuidema, Willem. Compositional Distributional Semantics with Long Short Term Memory. In *Proceedings of Joint Conference on Lexical and Computational Semantics (to appear)*, 2015.

Liu, Bing and Zhang, Lei. A survey of opinion mining and sentiment analysis. In *Mining Text Data*, pp. 415–463. 2012.

Liwicki, Marcus, Graves, Alex, Bunke, Horst, and Schmidhuber, Juergen. A novel approach to on-line handwriting recognition based on bidirectional long short-term memory networks. In *Proceedings of ICDAR*, 2007.

Manning, Christopher D. and Schütze, Hinrich. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, USA, 1999. ISBN 0-262-13360-1.

Mohammad, Saif M., Kiritchenko, Svetlana, and Martin, Joel. Identifying purpose behind electoral tweets. In *Proceedings of the 2nd International Workshop on Issues of Sentiment Discovery and Opinion Mining*, pp. 1–9, 2013.

Moilanen, Karo and Pulman, Stephen. Sentiment composition. In *Proceedings of RANLP*, Borovets, Bulgaria, 2007.

Pang, Bo and Lee, Lillian. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of ACL*, pp. 115–124, 2005.

Pang, Bo and Lee, Lillian. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1–2):1–135, 2008.

Pennington, Jeffrey, Socher, Richard, and Manning, Christopher. Glove: Global vectors for word representation. In *Proceedings of EMNLP*, August 2013.

Pinheiro, P. H. O. and Collobert, R. Recurrent convolutional neural networks for scene labeling. In *Proceedings of ICML*, 2014.

Socher, Richard, Lin, Cliff C., Ng, Andrew Y., and Manning, Christopher D. Parsing Natural Scenes and Natural Language with Recursive Neural Networks. In *Proceedings of ICML*, 2011.

Socher, Richard, Huval, Brody, Manning, Christopher D., and Ng, Andrew Y. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of EMNLP*, 2012.

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

Starzyk, Janusz A. and He, Haibo. Anticipation-based temporal sequences learning in hierarchical structure. *IEEE Transactions on Neural Networks*, 18(2), 2007.

Sutskever, Ilya, Vinyals, Oriol, and Le, Quoc V. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014.

Tai, Kai Sheng, Socher, Richard, and Manning, Christopher. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In *Proceedings of ACL (to appear)*, 2015.

Vinyals, Oriol, Toshev, Alexander, Bengio, Samy, and Erhan, Dumitru. Show and tell: A neural image caption generator. *CoRR*, abs/1411.4555, 2014.

Wilson, Theresa, Wiebe, Janyce, and Hoffmann, Paul. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of EMNLP*, Vancouver, Canada, 2005.

Zhu, Xiaodan, Guo, Hongyu, Mohammad, Saif, and Kiritchenko, Svetlana. An empirical study on the effect of negation words on sentiment. In *Proceedings of ACL*, Baltimore, Maryland, USA, June 2014a.

Zhu, Xiaodan, Kiritchenko, Svetlana, and Mohammad, Saif. NRC-Canada-2014: Recent Improvements in the Sentiment Analysis of Tweets. In *Proceedings of the International Workshop on Semantic Evaluation*, 2014b.