Long Short-Term Memory Over Tree 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

arXiv:1503.04881v1 [cs.CL] 16 Mar 2015

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 cell* in a *full-order* time sequence.

Recursion is a fundamental process associated with many problems—a recursive process and hierarchical structure so formed are common in different modalities. For example, semantics of sentences in human languages is believed to be carried by not merely a linear concatenation of words; instead, sentences have parse structures (Manning & Schütze, 1999). Image understanding, as another example, benefits 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 learn memory cells 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 interaction over trees. This is a desirable characteristic as many of such structures are deep. S-LSTM can be considered as bringing the merits of a recursive neural network and a recurrent neural network together[1]. In short, S-LSTM wires memory blocks in a partial-order 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 good representations for meaning of text is core to automatically understanding human languages. More specifically, we experiment with the models on the

---

On February 6th, 2015, this work was submitted to the International Conference on Machine Learning (ICML).

[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.

Stanford Sentiment Tree Bank (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 we propose here. We showed 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 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 & Kchler, 1996).

In addition, the literature has also included many other efforts of applying feedforward-based neural network over structures, including (Goller & Kchler, 1996; Chater, 1992; Starzyk et al.; Hammer et al., 2004), amongst others. For instance, Legrand and Collobert leverage neural networks over greedy syntactic parsing (Pinheiro & Collobert, 2014). In (Irsoy & Cardie, 2014), a deep recursive neural network is proposed . 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, 2012), 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, 2012), 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.

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 fig-

ure, the gray and light-blue leaf. In the figure, the small circle ("○") 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.
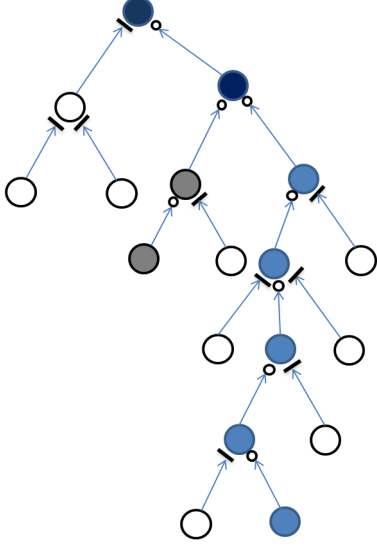

Figure 1. An example of S-LSTM, a long-short term memory network on tree structures. A tree node can consider information from multiple descendants. Information of the other nodes in white are blocked. The small circle ("○") or short line ("−") at each arrowhead indicates a *pass* or *block* of information, 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. Extension of the model to handle more children is rather straightforward.

As shown in the figure, the hidden vectors of the two children, denoted as h_{t-1}^L for the left child and h_{t-1}^R 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_{t-1}^L and h_{t-1}^R) and cell vectors (c_{t-1}^L and c_{t-1}^R) of its two children. These four sources of information are also used to form the gating signals for the left forget gate f_{t-1}^L and right forget gate f_{t-1}^R, where the weights used to combining them are specific to each of these gates, denoted as
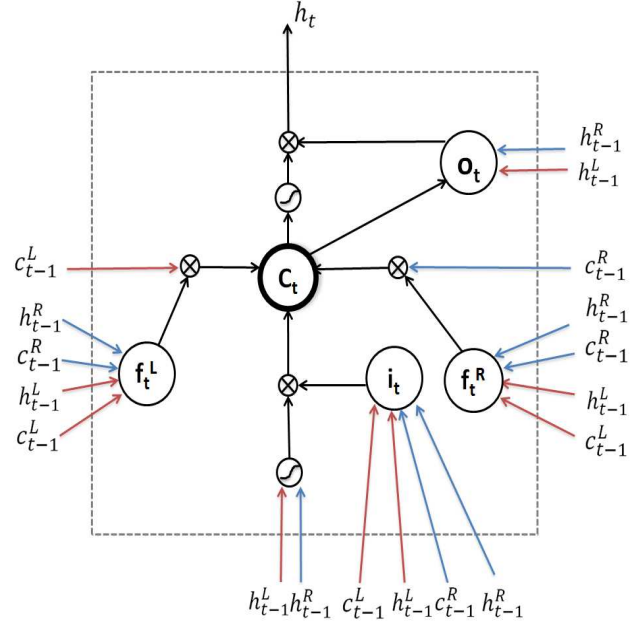

Figure 2. A S-LSTM memory block, consisting of an input gate, two forget gates, and an output gate. Hidden vectors h_{t-1}^* and cell vectors c_{t-1}^* from the left (red arrows) and right (blue arrows) children are deployed to compute c_t and h_t. $\otimes$ denotes a Hadamard product, and the "s" shaped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

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_{t-1}^L, c_{t-1}^R), 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.

$$i_t = \sigma(W_{hi}^L h_{t-1}^L + W_{hi}^R h_{t-1}^R + W_{ci}^L c_{t-1}^L$$
$$+ W_{ci}^R c_{t-1}^R + b_i) \quad (1)$$

$$f_t^L = \sigma(W_{hf_l}^L h_{t-1}^L + W_{hf_l}^R h_{t-1}^R + W_{cf_l}^L c_{t-1}^L$$
$$+ W_{cf_l}^R c_{t-1}^R + b_{f_l}) \tag{2}$$

$$f_t^R = \sigma(W_{hf_r}^L h_{t-1}^L + W_{hf_r}^R h_{t-1}^R + W_{cf_r}^L c_{t-1}^L$$
$$+ W_{cf_r}^R c_{t-1}^R + b_{f_r}) \tag{3}$$

$$x_t = W_{hx}^L h_{t-1}^L + W_{hx}^R h_{t-1}^R + b_x \tag{4}$$

$$c_t = f_t^L \otimes c_{t-1}^L + f_t^R \otimes c_{t-1}^R + i_t \otimes tanh(x_t) \tag{5}$$

$$o_t = \sigma(W_{ho}^L h_{t-1}^L + W_{ho}^R h_{t-1}^R + W_{co} c_t + b_o) \tag{6}$$

$$h_t = o_t \otimes \tanh(c_t) \tag{7}$$

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 & Kchler, 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 backprop formulas is tedious but we list them below to facilitate duplication of our work [2]. 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_t^h. The derivatives of the output gate δ_t^o, left forget gate $\delta_t^{f_l}$, right forget gate $\delta_t^{f_r}$, and input gate δ_t^i are computed as:

$$\epsilon_t^h = \frac{\partial O}{\partial h_t} \tag{8}$$

$$\delta_t^o = \epsilon_t^h \otimes \tanh(c_t) \otimes \sigma'(o_t) \tag{9}$$

$$\delta_t^{f_l} = \epsilon_t^c \otimes c_{t-1}^L \otimes \sigma'(f_t^L) \tag{10}$$

$$\delta_t^{f_r} = \epsilon_t^c \otimes c_{t-1}^R \otimes \sigma'(f_t^R) \tag{11}$$

$$\delta_t^i = \epsilon_t^c \otimes \tanh(x_t) \otimes \sigma'(i_t) \tag{12}$$

[2] The code will be published at www.icml-placeholder-only.com

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_t^c 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_t^c, otherwise Formula (14) is used:

$$\epsilon_t^c = \epsilon_t^h \otimes o_t \otimes g'(c_t) + \epsilon_{t+1}^c \otimes f_{t+1}^L +$$
$$(W_{ci}^L)^T \delta_{t+1}^i + (W_{cf_l}^L)^T \delta_{t+1}^{f_l} +$$
$$(W_{cf_r}^L)^T \delta_{t+1}^{f_r} + (W_{co}^L)^T \delta_t^o \tag{13}$$

$$\epsilon_t^c = \epsilon_t^h \otimes o_t \otimes g'(c_t) + \epsilon_{t+1}^c \otimes f_{t+1}^R +$$
$$(W_{ci}^R)^T \delta_{t+1}^i + (W_{cf_l}^R)^T \delta_{t+1}^{f_l} +$$
$$(W_{cf_r}^R)^T \delta_{t+1}^{f_r} + (W_{co}^R)^T \delta_t^o \tag{14}$$

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 correctness of the S-LSTM implementation with the standard approximated gradient approach.

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 Tree Bank 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 Tree Bank, 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 Tree Bank (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:

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

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 Tree Bank. 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) [3]. 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 Tree Bank, 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	**48.0**†	**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 words), or different initializations of word embedding, may further improve the performances reported here.

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). From these figures, we can observe that S-LSTM converge 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

[3]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.

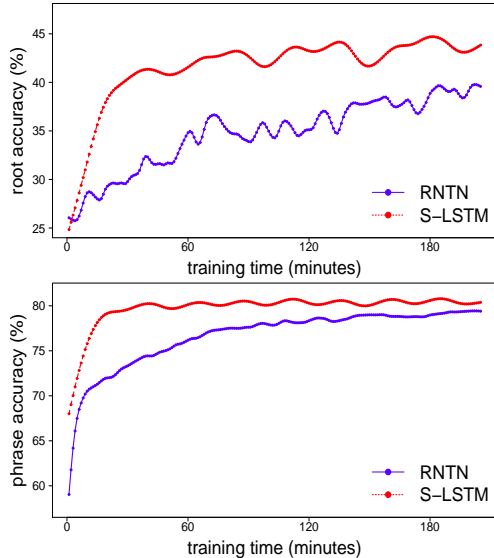much less parameters than RNTN and the forward and backward propagation can be computed efficiently.

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 2. *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"* [4]. Also, annotating all phrases is expensive. However, these should not be regarded as comments on the value of the Sentiment Tree Bank. Detailed annotations in the tree bank enable much interesting work to be possible, e.g., the study of the effect of negation in changing sentiment (Zhu et al., 2014).

The second setting, corresponding to model (3) and (4) in Table 2, 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) ($LEAF LBLS$ along the side of 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 between the two settings here.

---

[4]Phrase-level sentiment analysis is often defined over a very small subset of phrases of interest, such as in the phrase-level task defined in (Wilson et al., 2005; Mohammad et al., 2013).

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 2. 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†

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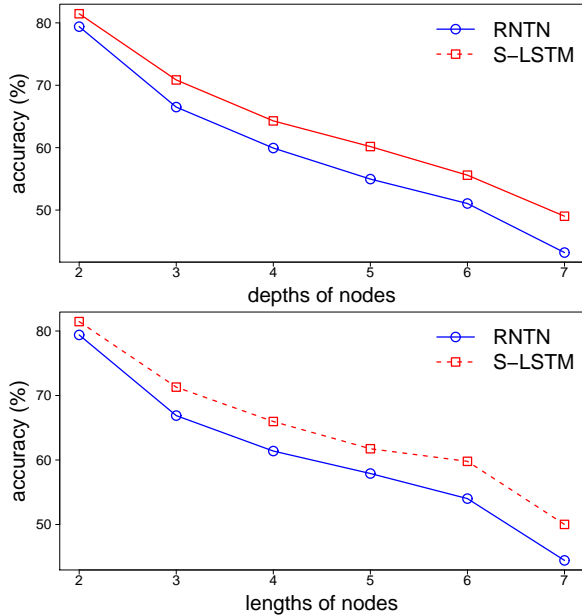
Figure 4. Accuracies at different depths (the upper figure) in the trees, or by different lengths of the phrases (the lower figure).

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 Tree Bank, 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 3 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 are better. Note that in real applications, where there is no 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.

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

Table 3. 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†

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 recurrent neural networks can actually capture the structures implicitly through a linear-chain 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 *in: proceedings of the fourteenth annual conference of the cognitive science society. (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 the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pp. 793–801, Honolulu, Hawaii, 2008.

Eck, Douglas and Schmidhuber, Jrgen. Finding temporal structure in music: Blues improvisation with lstm recurrent networks. In *NEURAL NETWORKS FOR SIGNAL PROCESSING XII, PROCEEDINGS OF THE 2002 IEEE WORKSHOP*, pp. 747–756. IEEE, 2002a.

Eck, Douglas and Schmidhuber, Jrgen. Learning the long-term structure of the blues. In *IN PROC. INTL. CONF*, pp. 284–289, 2002b.

Goller, Christoph and Kchler, Andreas. Learning task-dependent distributed representations by backpropagation through structure. In *In Proc. of the ICNN-96*, pp. 347–352, Bochum, Germany, 1996. IEEE.

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

Graves, Alex. *Supervised sequence labelling with recurrent neural networks*, volume 385. Springer, 2012.

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

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

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 Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., and Weinberger, K.Q. (eds.), *Advances in Neural Information Processing Systems 27*, pp. 2096–2104. Curran Associates, Inc., 2014.

Kalchbrenner, Nal, Grefenstette, Edward, and Blunsom, Phil. A convolutional neural network for modelling sentences. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, June 2014.

Klein, Dan and Manning, Christopher D. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pp. 423–430, Sapporo, Japan, 2003. Association for Computational Linguistics. doi: 10.3115/1075096.1075150.

Liu, Bing and Zhang, Lei. A survey of opinion mining and sentiment analysis. In Aggarwal, Charu C. and Zhai, ChengXiang (eds.), *Mining Text Data*, pp. 415–463. Springer US, 2012. ISBN 978-1-4614-3222-7. doi: 10.1007/978-1-4614-3223-4_13.

Liwicki, Marcus, Graves, Alex, Bunke, Horst, and Schmidhuber, Jrgen. A novel approach to on-line handwriting recognition based on bidirectional long short-term memory networks. In *In Proceedings of the 9th International Conference on Document Analysis and Recognition, ICDAR 2007*, 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*, WISDOM '13, pp. 1–9, 2013.

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

Pang, Bo and Lee, Lillian. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, ACL '05, 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.

Pinheiro, P. H. O. and Collobert, R. Recurrent convolutional neural networks for scene labeling. In *Proceedings of the 31st International Conference on Machine Learning (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 the 26th International Conference on Machine Learning (ICML)*, 2011.

Socher, Richard, Huval, Brody, Manning, Christopher D., and Ng, Andrew Y. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '12, Jeju, Korea, 2012. Association for Computational Linguistics.

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 the Conference on Empirical Methods in Natural Language Processing*, EMNLP '13, Seattle, USA, 2013. Association for Computational Linguistics.

Starzyk, Janusz A., Member, Senior, and He, Haibo. Anticipation-based temporal sequences learning in hierarchical structure.

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

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 the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pp. 347–354, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. doi: 10.3115/1220575.1220619.

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 2014.