Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks

Kai Sheng Tai, Richard Socher*, Christopher D. Manning
Computer Science Department, Stanford University, *MetaMind Inc.
kst@cs.stanford.edu, richard@metamind.io, manning@stanford.edu

Abstract

Because of their superior ability to preserve sequence information over time, Long Short-Term Memory (LSTM) networks, a type of recurrent neural network with a more complex computational unit, have obtained strong results on a variety of sequence modeling tasks. The only underlying LSTM structure that has been explored so far is a linear chain. However, natural language exhibits syntactic properties that would naturally combine words to phrases. We introduce the Tree-LSTM, a generalization of LSTMs to tree-structured network topologies. Tree-LSTMs outperform all existing systems and strong LSTM baselines on two tasks: predicting the semantic relatedness of two sentences (SemEval 2014, Task 1) and sentiment classification (Stanford Sentiment Treebank).

1 Introduction

Most models for distributed representations of phrases and sentences—that is, models where real-valued vectors are used to represent meaning—fall into one of three classes: bag-of-words models, sequence models, and tree-structured models. In bag-of-words models, phrase and sentence representations are independent of word order; for example, they can be generated by averaging constituent word representations (Landauer and Dumais, 1997; Foltz et al., 1998). In contrast, sequence models construct sentence representations as an order-sensitive function of the sequence of tokens (Elman, 1990; Mikolov, 2012). Lastly, tree-structured models compose each phrase and sentence representation from its constituent subphrases according to a given syntactic structure over the sentence (Goller and Kuchler, 1996; Socher et al., 2011).

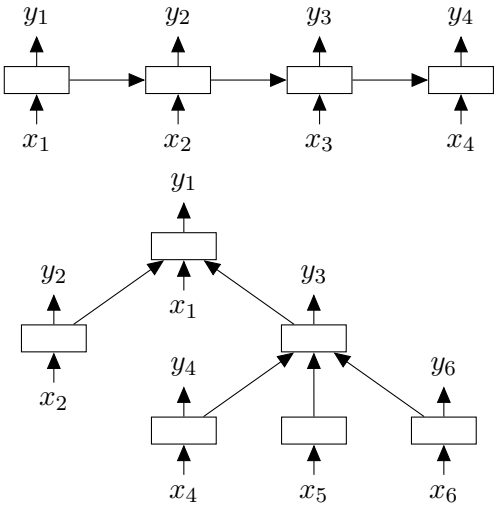

Figure 1: **Top:** A chain-structured LSTM network. **Bottom:** A tree-structured LSTM network with arbitrary branching factor.

Order-insensitive models are insufficient to fully capture the semantics of natural language due to their inability to account for differences in meaning as a result of differences in word order or syntactic structure (*e.g.,* "cats climb trees" *vs.* "trees climb cats"). We therefore turn to order-sensitive sequential or tree-structured models. In particular, tree-structured models are a linguistically attractive option due to their relation to syntactic interpretations of sentence structure. A natural question, then, is the following: to what extent (if at all) can we do better with tree-structured models as opposed to sequential models for sentence representation? In this paper, we work towards addressing this question by directly comparing a type of sequential model that has recently been used to achieve state-of-the-art results in several NLP tasks against its tree-structured generalization.

Due to their capability for processing arbitrary-length sequences, recurrent neural networks

Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pages 1556–1566, Beijing, China, July 26-31, 2015.

(RNNs) are a natural choice for sequence modeling tasks. Recently, RNNs with Long Short-Term Memory (LSTM) units (Hochreiter and Schmidhuber, 1997) have re-emerged as a popular architecture due to their representational power and effectiveness at capturing long-term dependencies. LSTM networks, which we review in Sec. 2, have been successfully applied to a variety of sequence modeling and prediction tasks, notably machine translation (Bahdanau et al., 2015; Sutskever et al., 2014), speech recognition (Graves et al., 2013), image caption generation (Vinyals et al., 2014), and program execution (Zaremba and Sutskever, 2014).

In this paper, we introduce a generalization of the standard LSTM architecture to tree-structured network topologies and show its superiority for representing sentence meaning over a sequential LSTM. While the standard LSTM composes its hidden state from the input at the current time step and the hidden state of the LSTM unit in the previous time step, the tree-structured LSTM, or Tree-LSTM, composes its state from an input vector and the hidden states of arbitrarily many child units. The standard LSTM can then be considered a special case of the Tree-LSTM where each internal node has exactly one child.

In our evaluations, we demonstrate the empirical strength of Tree-LSTMs as models for representing sentences. We evaluate the Tree-LSTM architecture on two tasks: semantic relatedness prediction on sentence pairs and sentiment classification of sentences drawn from movie reviews. Our experiments show that Tree-LSTMs outperform existing systems and sequential LSTM baselines on both tasks. Implementations of our models and experiments are available at `https://github.com/stanfordnlp/treelstm`.

2 Long Short-Term Memory Networks

2.1 Overview

Recurrent neural networks (RNNs) are able to process input sequences of arbitrary length via the recursive application of a transition function on a *hidden state vector* h_t. At each time step t, the hidden state h_t is a function of the input vector x_t that the network receives at time t and its previous hidden state h_{t-1}. For example, the input vector x_t could be a vector representation of the t-th word in body of text (Elman, 1990; Mikolov, 2012). The hidden state $h_t \in \mathbb{R}^d$ can be interpreted as a d-dimensional distributed representation of the sequence of tokens observed up to time t.

Commonly, the RNN transition function is an affine transformation followed by a pointwise nonlinearity such as the hyperbolic tangent function:

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

Unfortunately, a problem with RNNs with transition functions of this form is that during training, components of the gradient vector can grow or decay exponentially over long sequences (Hochreiter, 1998; Bengio et al., 1994). This problem with *exploding* or *vanishing gradients* makes it difficult for the RNN model to learn long-distance correlations in a sequence.

The LSTM architecture (Hochreiter and Schmidhuber, 1997) addresses this problem of learning long-term dependencies by introducing a *memory cell* that is able to preserve state over long periods of time. While numerous LSTM variants have been described, here we describe the version used by Zaremba and Sutskever (2014).

We define the LSTM *unit* at each time step t to be a collection of vectors in $\mathbb{R}^d$: an *input gate* i_t, a *forget gate* f_t, an *output gate* o_t, a *memory cell* c_t and a hidden state h_t. The entries of the gating vectors i_t, f_t and o_t are in $[0, 1]$. We refer to d as the *memory dimension* of the LSTM.

The LSTM transition equations are the following:

$$\begin{aligned}
i_t &= \sigma\left(W^{(i)}x_t + U^{(i)}h_{t-1} + b^{(i)}\right), \qquad (1)\\
f_t &= \sigma\left(W^{(f)}x_t + U^{(f)}h_{t-1} + b^{(f)}\right),\\
o_t &= \sigma\left(W^{(o)}x_t + U^{(o)}h_{t-1} + b^{(o)}\right),\\
u_t &= \tanh\left(W^{(u)}x_t + U^{(u)}h_{t-1} + b^{(u)}\right),\\
c_t &= i_t \odot u_t + f_t \odot c_{t-1},\\
h_t &= o_t \odot \tanh(c_t),
\end{aligned}$$

where x_t is the input at the current time step, σ denotes the logistic sigmoid function and $\odot$ denotes elementwise multiplication. Intuitively, the forget gate controls the extent to which the previous memory cell is forgotten, the input gate controls how much each unit is updated, and the output gate controls the exposure of the internal memory state. The hidden state vector in an LSTM unit is therefore a gated, partial view of the state of the unit's internal memory cell. Since the value of the gating variables vary for each vector element, the model

can learn to represent information over multiple time scales.

2.2 Variants

Two commonly-used variants of the basic LSTM architecture are the Bidirectional LSTM and the Multilayer LSTM (also known as the *stacked* or *deep* LSTM).

Bidirectional LSTM. A Bidirectional LSTM (Graves et al., 2013) consists of two LSTMs that are run in parallel: one on the input sequence and the other on the reverse of the input sequence. At each time step, the hidden state of the Bidirectional LSTM is the concatenation of the forward and backward hidden states. This setup allows the hidden state to capture both past and future information.

Multilayer LSTM. In Multilayer LSTM architectures, the hidden state of an LSTM unit in layer ℓ is used as input to the LSTM unit in layer $\ell+1$ in the same time step (Graves et al., 2013; Sutskever et al., 2014; Zaremba and Sutskever, 2014). Here, the idea is to let the higher layers capture longer-term dependencies of the input sequence.

These two variants can be combined as a Multilayer Bidirectional LSTM (Graves et al., 2013).

3 Tree-Structured LSTMs

A limitation of the LSTM architectures described in the previous section is that they only allow for strictly sequential information propagation. Here, we propose two natural extensions to the basic LSTM architecture: the *Child-Sum Tree-LSTM* and the *N-ary Tree-LSTM*. Both variants allow for richer network topologies where each LSTM unit is able to incorporate information from multiple child units.

As in standard LSTM units, each Tree-LSTM unit (indexed by j) contains input and output gates i_j and o_j, a memory cell c_j and hidden state h_j. The difference between the standard LSTM unit and Tree-LSTM units is that gating vectors and memory cell updates are dependent on the states of possibly many child units. Additionally, instead of a single forget gate, the Tree-LSTM unit contains one forget gate f_{jk} for each child k. This allows the Tree-LSTM unit to selectively incorporate information from each child. For example, a Tree-LSTM model can learn to emphasize semantic heads in a semantic relatedness task, or it can learn to preserve the representation of sentiment-rich children for sentiment classification.

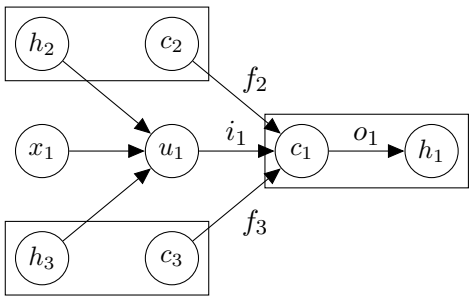

Figure 2: Composing the memory cell c_1 and hidden state h_1 of a Tree-LSTM unit with two children (subscripts 2 and 3). Labeled edges correspond to gating by the indicated gating vector, with dependencies omitted for compactness.

As with the standard LSTM, each Tree-LSTM unit takes an input vector x_j. In our applications, each x_j is a vector representation of a word in a sentence. The input word at each node depends on the tree structure used for the network. For instance, in a Tree-LSTM over a dependency tree, each node in the tree takes the vector corresponding to the head word as input, whereas in a Tree-LSTM over a constituency tree, the leaf nodes take the corresponding word vectors as input.

3.1 Child-Sum Tree-LSTMs

Given a tree, let $C(j)$ denote the set of children of node j. The Child-Sum Tree-LSTM transition equations are the following:

$$\tilde{h}_j = \sum_{k \in C(j)} h_k, \tag{2}$$

$$i_j = \sigma\left(W^{(i)} x_j + U^{(i)} \tilde{h}_j + b^{(i)}\right), \tag{3}$$

$$f_{jk} = \sigma\left(W^{(f)} x_j + U^{(f)} h_k + b^{(f)}\right), \tag{4}$$

$$o_j = \sigma\left(W^{(o)} x_j + U^{(o)} \tilde{h}_j + b^{(o)}\right), \tag{5}$$

$$u_j = \tanh\left(W^{(u)} x_j + U^{(u)} \tilde{h}_j + b^{(u)}\right), \tag{6}$$

$$c_j = i_j \odot u_j + \sum_{k \in C(j)} f_{jk} \odot c_k, \tag{7}$$

$$h_j = o_j \odot \tanh(c_j), \tag{8}$$

where in Eq. 4, $k \in C(j)$.

Intuitively, we can interpret each parameter matrix in these equations as encoding correlations between the component vectors of the Tree-LSTM

unit, the input x_j, and the hidden states h_k of the unit's children. For example, in a dependency tree application, the model can learn parameters $W^{(i)}$ such that the components of the input gate i_j have values close to 1 (*i.e.*, "open") when a semantically important content word (such as a verb) is given as input, and values close to 0 (*i.e.*, "closed") when the input is a relatively unimportant word (such as a determiner).

Dependency Tree-LSTMs. Since the Child-Sum Tree-LSTM unit conditions its components on the sum of child hidden states h_k, it is well-suited for trees with high branching factor or whose children are unordered. For example, it is a good choice for dependency trees, where the number of dependents of a head can be highly variable. We refer to a Child-Sum Tree-LSTM applied to a dependency tree as a *Dependency Tree-LSTM*.

3.2 N-ary Tree-LSTMs

The N-ary Tree-LSTM can be used on tree structures where the branching factor is at most N and where children are ordered, *i.e.*, they can be indexed from 1 to N. For any node j, write the hidden state and memory cell of its kth child as h_{jk} and c_{jk} respectively. The N-ary Tree-LSTM transition equations are the following:

$$i_j = \sigma\left(W^{(i)}x_j + \sum_{\ell=1}^{N} U_\ell^{(i)} h_{j\ell} + b^{(i)}\right), \quad (9)$$

$$f_{jk} = \sigma\left(W^{(f)}x_j + \sum_{\ell=1}^{N} U_{k\ell}^{(f)} h_{j\ell} + b^{(f)}\right), \quad (10)$$

$$o_j = \sigma\left(W^{(o)}x_j + \sum_{\ell=1}^{N} U_\ell^{(o)} h_{j\ell} + b^{(o)}\right), \quad (11)$$

$$u_j = \tanh\left(W^{(u)}x_j + \sum_{\ell=1}^{N} U_\ell^{(u)} h_{j\ell} + b^{(u)}\right), \quad (12)$$

$$c_j = i_j \odot u_j + \sum_{\ell=1}^{N} f_{j\ell} \odot c_{j\ell}, \quad (13)$$

$$h_j = o_j \odot \tanh(c_j), \quad (14)$$

where in Eq. 10, $k = 1, 2, \ldots, N$. Note that when the tree is simply a chain, both Eqs. 2–8 and Eqs. 9–14 reduce to the standard LSTM transitions, Eqs. 1.

The introduction of separate parameter matrices for each child k allows the N-ary Tree-LSTM model to learn more fine-grained conditioning on the states of a unit's children than the Child-Sum Tree-LSTM. Consider, for example, a constituency tree application where the left child of a node corresponds to a noun phrase, and the right child to a verb phrase. Suppose that in this case it is advantageous to emphasize the verb phrase in the representation. Then the $U_{k\ell}^{(f)}$ parameters can be trained such that the components of f_{j1} are close to 0 (*i.e.*, "forget"), while the components of f_{j2} are close to 1 (*i.e.*, "preserve").

Forget gate parameterization. In Eq. 10, we define a parameterization of the kth child's forget gate f_{jk} that contains "off-diagonal" parameter matrices $U_{k\ell}^{(f)}$, $k \neq \ell$. This parameterization allows for more flexible control of information propagation from child to parent. For example, this allows the left hidden state in a binary tree to have either an *excitatory* or *inhibitory* effect on the forget gate of the right child. However, for large values of N, these additional parameters are impractical and may be tied or fixed to zero.

Constituency Tree-LSTMs. We can naturally apply *Binary* Tree-LSTM units to binarized constituency trees since left and right child nodes are distinguished. We refer to this application of Binary Tree-LSTMs as a *Constituency Tree-LSTM*. Note that in Constituency Tree-LSTMs, a node j receives an input vector x_j only if it is a leaf node.

In the remainder of this paper, we focus on the special cases of Dependency Tree-LSTMs and Constituency Tree-LSTMs. These architectures are in fact closely related; since we consider only binarized constituency trees, the parameterizations of the two models are very similar. The key difference is in the application of the compositional parameters: dependent *vs.* head for Dependency Tree-LSTMs, and left child *vs.* right child for Constituency Tree-LSTMs.

4 Models

We now describe two specific models that apply the Tree-LSTM architectures described in the previous section.

4.1 Tree-LSTM Classification

In this setting, we wish to predict labels $\hat{y}$ from a discrete set of classes $\mathcal{Y}$ for some subset of nodes in a tree. For example, the label for a node in a

parse tree could correspond to some property of the phrase spanned by that node.

At each node j, we use a softmax classifier to predict the label $\hat{y}_j$ given the inputs $\{x\}_j$ observed at nodes in the subtree rooted at j. The classifier takes the hidden state h_j at the node as input:

$$\hat{p}_\theta(y \mid \{x\}_j) = \text{softmax}\left(W^{(s)} h_j + b^{(s)}\right),$$
$$\hat{y_j} = \arg\max_y \hat{p}_\theta\left(y \mid \{x\}_j\right).$$

The cost function is the negative log-likelihood of the true class labels $y^{(k)}$ at each labeled node:

$$J(\theta) = -\frac{1}{m}\sum_{k=1}^{m} \log \hat{p}_\theta\Big(y^{(k)} \;\Big|\; \{x\}^{(k)}\Big) + \frac{\lambda}{2}\|\theta\|_2^2,$$

where m is the number of labeled nodes in the training set, the superscript k indicates the kth labeled node, and λ is an L2 regularization hyperparameter.

4.2 Semantic Relatedness of Sentence Pairs

Given a sentence pair, we wish to predict a real-valued similarity score in some range $[1, K]$, where $K > 1$ is an integer. The sequence $\{1, 2, \ldots, K\}$ is some ordinal scale of similarity, where higher scores indicate greater degrees of similarity, and we allow real-valued scores to account for ground-truth ratings that are an average over the evaluations of several human annotators.

We first produce sentence representations h_L and h_R for each sentence in the pair using a Tree-LSTM model over each sentence's parse tree. Given these sentence representations, we predict the similarity score $\hat{y}$ using a neural network that considers both the distance and angle between the pair (h_L, h_R):

$$\begin{aligned} h_\times &= h_L \odot h_R, \\ h_+ &= |h_L - h_R|, \\ h_s &= \sigma\left(W^{(\times)} h_\times + W^{(+)} h_+ + b^{(h)}\right), \\ \hat{p}_\theta &= \text{softmax}\left(W^{(p)} h_s + b^{(p)}\right), \\ \hat{y} &= r^T \hat{p}_\theta, \end{aligned} \tag{15}$$

where $r^T = [1\ 2\ \ldots\ K]$ and the absolute value function is applied elementwise. The use of both distance measures $h_\times$ and h_+ is empirically motivated: we find that the combination outperforms the use of either measure alone. The multiplicative measure $h_\times$ can be interpreted as an elementwise comparison of the signs of the input representations.

We want the expected rating under the predicted distribution $\hat{p}_\theta$ given model parameters θ to be close to the gold rating $y \in [1, K]$: $\hat{y} = r^T \hat{p}_\theta \approx y$. We therefore define a sparse target distribution[1] p that satisfies $y = r^T p$:

$$p_i = \begin{cases} y - \lfloor y \rfloor, & i = \lfloor y \rfloor + 1 \\ \lfloor y \rfloor - y + 1, & i = \lfloor y \rfloor \\ 0 & \text{otherwise} \end{cases}$$

for $1 \le i \le K$. The cost function is the regularized KL-divergence between p and $\hat{p}_\theta$:

$$J(\theta) = \frac{1}{m}\sum_{k=1}^{m} \text{KL}\Big(p^{(k)} \;\Big\|\; \hat{p}_\theta^{(k)}\Big) + \frac{\lambda}{2}\|\theta\|_2^2,$$

where m is the number of training pairs and the superscript k indicates the kth sentence pair.

5 Experiments

We evaluate our Tree-LSTM architectures on two tasks: (1) sentiment classification of sentences sampled from movie reviews and (2) predicting the semantic relatedness of sentence pairs.

In comparing our Tree-LSTMs against sequential LSTMs, we control for the number of LSTM parameters by varying the dimensionality of the hidden states[2]. Details for each model variant are summarized in Table 1.

5.1 Sentiment Classification

In this task, we predict the sentiment of sentences sampled from movie reviews. We use the Stanford Sentiment Treebank (Socher et al., 2013). There are two subtasks: binary classification of sentences, and fine-grained classification over five classes: very negative, negative, neutral, positive, and very positive. We use the standard train/dev/test splits of 6920/872/1821 for the binary classification subtask and 8544/1101/2210 for the fine-grained classification subtask (there are fewer examples for the binary subtask since

[1] In the subsequent experiments, we found that optimizing this objective yielded better performance than a mean squared error objective.

[2] For our Bidirectional LSTMs, the parameters of the forward and backward transition functions are shared. In our experiments, this achieved superior performance to Bidirectional LSTMs with untied weights and the same number of parameters (and therefore smaller hidden vector dimensionality).

	Relatedness		Sentiment	
LSTM Variant	d	$\|\theta\|$	d	$\|\theta\|$
Standard	150	203,400	168	315,840
Bidirectional	150	203,400	168	315,840
2-layer	108	203,472	120	318,720
Bidirectional 2-layer	108	203,472	120	318,720
Constituency Tree	142	205,190	150	316,800
Dependency Tree	150	203,400	168	315,840

Table 1: Memory dimensions d and composition function parameter counts $|\theta|$ for each LSTM variant that we evaluate.

neutral sentences are excluded). Standard binarized constituency parse trees are provided for each sentence in the dataset, and each node in these trees is annotated with a sentiment label.

For the sequential LSTM baselines, we predict the sentiment of a phrase using the representation given by the final LSTM hidden state. The sequential LSTM models are trained on the spans corresponding to labeled nodes in the training set.

We use the classification model described in Sec. 4.1 with both Dependency Tree-LSTMs (Sec. 3.1) and Constituency Tree-LSTMs (Sec. 3.2). The Constituency Tree-LSTMs are structured according to the provided parse trees. For the Dependency Tree-LSTMs, we produce dependency parses[3] of each sentence; each node in a tree is given a sentiment label if its span matches a labeled span in the training set.

5.2 Semantic Relatedness

For a given pair of sentences, the semantic relatedness task is to predict a human-generated rating of the similarity of the two sentences in meaning.

We use the Sentences Involving Compositional Knowledge (SICK) dataset (Marelli et al., 2014), consisting of 9927 sentence pairs in a 4500/500/4927 train/dev/test split. The sentences are derived from existing image and video description datasets. Each sentence pair is annotated with a relatedness score $y \in [1, 5]$, with 1 indicating that the two sentences are completely unrelated, and 5 indicating that the two sentences are very related. Each label is the average of 10 ratings assigned by different human annotators.

Here, we use the similarity model described in Sec. 4.2. For the similarity prediction network (Eqs. 15) we use a hidden layer of size 50. We

[3]Dependency parses produced by the Stanford Neural Network Dependency Parser (Chen and Manning, 2014).

Method	**Fine-grained**	**Binary**
RAE (Socher et al., 2013)	43.2	82.4
MV-RNN (Socher et al., 2013)	44.4	82.9
RNTN (Socher et al., 2013)	45.7	85.4
DCNN (Blunsom et al., 2014)	48.5	86.8
Paragraph-Vec (Le and Mikolov, 2014)	48.7	87.8
CNN-non-static (Kim, 2014)	48.0	87.2
CNN-multichannel (Kim, 2014)	47.4	**88.1**
DRNN (Irsoy and Cardie, 2014)	49.8	86.6
LSTM	46.4 (1.1)	84.9 (0.6)
Bidirectional LSTM	49.1 (1.0)	87.5 (0.5)
2-layer LSTM	46.0 (1.3)	86.3 (0.6)
2-layer Bidirectional LSTM	48.5 (1.0)	87.2 (1.0)
Dependency Tree-LSTM	48.4 (0.4)	85.7 (0.4)
Constituency Tree-LSTM		
– randomly initialized vectors	43.9 (0.6)	82.0 (0.5)
– Glove vectors, fixed	49.7 (0.4)	87.5 (0.8)
– Glove vectors, tuned	**51.0** (0.5)	88.0 (0.3)

Table 2: Test set accuracies on the Stanford Sentiment Treebank. For our experiments, we report mean accuracies over 5 runs (standard deviations in parentheses). **Fine-grained:** 5-class sentiment classification. **Binary:** positive/negative sentiment classification.

produce binarized constituency parses[4] and dependency parses of the sentences in the dataset for our Constituency Tree-LSTM and Dependency Tree-LSTM models.

5.3 Hyperparameters and Training Details

The hyperparameters for our models were tuned on the development set for each task.

We initialized our word representations using publicly available 300-dimensional Glove vectors[5] (Pennington et al., 2014). For the sentiment classification task, word representations were updated during training with a learning rate of 0.1. For the semantic relatedness task, word representations were held fixed as we did not observe any significant improvement when the representations were tuned.

Our models were trained using AdaGrad (Duchi et al., 2011) with a learning rate of 0.05 and a minibatch size of 25. The model parameters were regularized with a per-minibatch L2 regularization strength of 10^{-4}. The sentiment classifier was additionally regularized using dropout (Srivastava et al., 2014) with a dropout rate of 0.5. We did not observe performance gains using dropout on the semantic relatedness task.

[4]Constituency parses produced by the Stanford PCFG Parser (Klein and Manning, 2003).

[5]Trained on 840 billion tokens of Common Crawl data, `http://nlp.stanford.edu/projects/glove/`.

Method	Pearson's r	Spearman's ρ	MSE
Illinois-LH (Lai and Hockenmaier, 2014)	0.7993	0.7538	0.3692
UNAL-NLP (Jimenez et al., 2014)	0.8070	0.7489	0.3550
Meaning Factory (Bjerva et al., 2014)	0.8268	0.7721	0.3224
ECNU (Zhao et al., 2014)	0.8414	–	–
Mean vectors	0.7577 (0.0013)	0.6738 (0.0027)	0.4557 (0.0090)
DT-RNN (Socher et al., 2014)	0.7923 (0.0070)	0.7319 (0.0071)	0.3822 (0.0137)
SDT-RNN (Socher et al., 2014)	0.7900 (0.0042)	0.7304 (0.0076)	0.3848 (0.0074)
LSTM	0.8528 (0.0031)	0.7911 (0.0059)	0.2831 (0.0092)
Bidirectional LSTM	0.8567 (0.0028)	0.7966 (0.0053)	0.2736 (0.0063)
2-layer LSTM	0.8515 (0.0066)	0.7896 (0.0088)	0.2838 (0.0150)
2-layer Bidirectional LSTM	0.8558 (0.0014)	0.7965 (0.0018)	0.2762 (0.0020)
Constituency Tree-LSTM	0.8582 (0.0038)	0.7966 (0.0053)	0.2734 (0.0108)
Dependency Tree-LSTM	**0.8676** (0.0030)	**0.8083** (0.0042)	**0.2532** (0.0052)

Table 3: Test set results on the SICK semantic relatedness subtask. For our experiments, we report mean scores over 5 runs (standard deviations in parentheses). Results are grouped as follows: **(1)** SemEval 2014 submissions; **(2)** Our own baselines; **(3)** Sequential LSTMs; **(4)** Tree-structured LSTMs.

6 Results

6.1 Sentiment Classification

Our results are summarized in Table 2. The Constituency Tree-LSTM outperforms existing systems on the fine-grained classification subtask and achieves accuracy comparable to the state-of-the-art on the binary subtask. In particular, we find that it outperforms the Dependency Tree-LSTM. This performance gap is at least partially attributable to the fact that the Dependency Tree-LSTM is trained on less data: about 150K labeled nodes *vs.* 319K for the Constituency Tree-LSTM. This difference is due to (1) the dependency representations containing fewer nodes than the corresponding constituency representations, and (2) the inability to match about 9% of the dependency nodes to a corresponding span in the training data.

We found that updating the word representations during training ("fine-tuning" the word embedding) yields a significant boost in performance on the fine-grained classification subtask and gives a minor gain on the binary classification subtask (this finding is consistent with previous work on this task by Kim (2014)). These gains are to be expected since the Glove vectors used to initialize our word representations were not originally trained to capture sentiment.

6.2 Semantic Relatedness

Our results are summarized in Table 3. Following Marelli et al. (2014), we use Pearson's r, Spearman's ρ and mean squared error (MSE) as evaluation metrics. The first two metrics are measures of correlation against human evaluations of semantic relatedness.

We compare our models against a number of non-LSTM baselines. The mean vector baseline computes sentence representations as a mean of the representations of the constituent words. The DT-RNN and SDT-RNN models (Socher et al., 2014) both compose vector representations for the nodes in a dependency tree as a sum over affine-transformed child vectors, followed by a nonlinearity. The SDT-RNN is an extension of the DT-RNN that uses a separate transformation for each dependency relation. For each of our baselines, including the LSTM models, we use the similarity model described in Sec. 4.2.

We also compare against four of the top-performing systems[6] submitted to the SemEval 2014 semantic relatedness shared task: ECNU (Zhao et al., 2014), The Meaning Factory (Bjerva et al., 2014), UNAL-NLP (Jimenez et al., 2014), and Illinois-LH (Lai and Hockenmaier, 2014). These systems are heavily feature engineered, generally using a combination of surface form overlap features and lexical distance features derived from WordNet or the Paraphrase Database (Ganitkevitch et al., 2013).

Our LSTM models outperform all these sys-

[6]We list the strongest results we were able to find for this task; in some cases, these results are stronger than the official performance by the team on the shared task. For example, the listed result by Zhao et al. (2014) is stronger than their submitted system's Pearson correlation score of 0.8280.

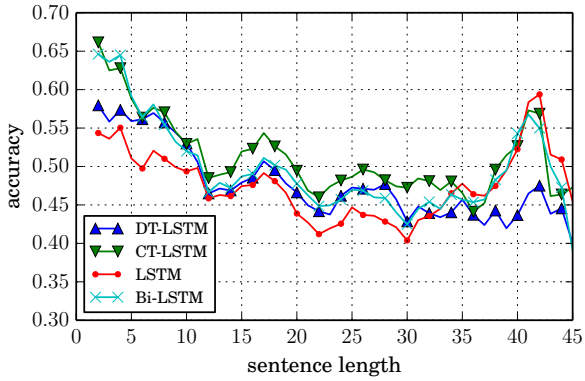

Figure 3: Fine-grained sentiment classification accuracy *vs.* sentence length. For each ℓ, we plot accuracy for the test set sentences with length in the window $[\ell - 2, \ell + 2]$. Examples in the tail of the length distribution are batched in the final window ($\ell = 45$).

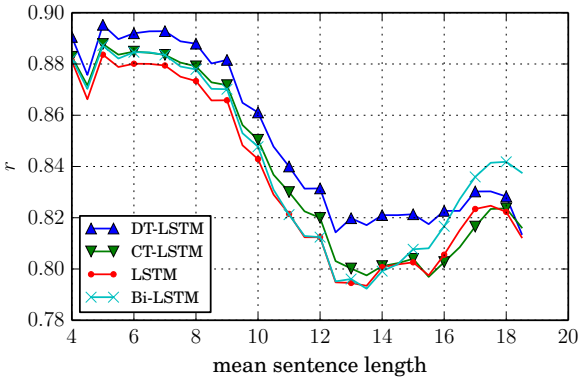

Figure 4: Pearson correlations r between predicted similarities and gold ratings *vs.* sentence length. For each ℓ, we plot r for the pairs with mean length in the window $[\ell-2, \ell+2]$. Examples in the tail of the length distribution are batched in the final window ($\ell = 18.5$).

tems without any additional feature engineering, with the best results achieved by the Dependency Tree-LSTM. Recall that in this task, both Tree-LSTM models only receive supervision at the root of the tree, in contrast to the sentiment classification task where supervision was also provided at the intermediate nodes. We conjecture that in this setting, the Dependency Tree-LSTM benefits from its more compact structure relative to the Constituency Tree-LSTM, in the sense that paths from input word vectors to the root of the tree are shorter on aggregate for the Dependency Tree-LSTM.

7 Discussion and Qualitative Analysis

7.1 Modeling Semantic Relatedness

In Table 4, we list nearest-neighbor sentences retrieved from a 1000-sentence sample of the SICK test set. We compare the neighbors ranked by the Dependency Tree-LSTM model against a baseline ranking by cosine similarity of the mean word vectors for each sentence.

The Dependency Tree-LSTM model exhibits several desirable properties. Note that in the dependency parse of the second query sentence, the word "ocean" is the second-furthest word from the root ("waving"), with a depth of 4. Regardless, the retrieved sentences are all semantically related to the word "ocean", which indicates that the Tree-LSTM is able to both preserve and emphasize information from relatively distant nodes. Additionally, the Tree-LSTM model shows greater robustness to differences in sentence length. Given the query "two men are playing guitar", the Tree-LSTM associates the phrase "playing guitar" with the longer, related phrase "dancing and singing in front of a crowd" (note as well that there is zero token overlap between the two phrases).

7.2 Effect of Sentence Length

One hypothesis to explain the empirical strength of Tree-LSTMs is that tree structures help mitigate the problem of preserving state over long sequences of words. If this were true, we would expect to see the greatest improvement over sequential LSTMs on longer sentences. In Figs. 3 and 4, we show the relationship between sentence length and performance as measured by the relevant task-specific metric. Each data point is a mean score over 5 runs, and error bars have been omitted for clarity.

We observe that while the Dependency Tree-LSTM does significantly outperform its sequential counterparts on the relatedness task for longer sentences of length 13 to 15 (Fig. 4), it also achieves consistently strong performance on shorter sentences. This suggests that unlike sequential LSTMs, Tree-LSTMs are able to encode semantically-useful structural information in the sentence representations that they compose.

8 Related Work

Distributed representations of words (Rumelhart et al., 1988; Collobert et al., 2011; Turian et al., 2010; Huang et al., 2012; Mikolov et al., 2013;

Ranking by mean word vector cosine similarity	Score
a woman is slicing potatoes	
a woman is cutting potatoes	0.96
a woman is slicing herbs	0.92
a woman is slicing tofu	0.92
a boy is waving at some young runners from the ocean	
a man and a boy are standing at the bottom of some stairs , which are outdoors	0.92
a group of children in uniforms is standing at a gate and one is kissing the mother	0.90
a group of children in uniforms is standing at a gate and there is no one kissing the mother	0.90
two men are playing guitar	
some men are playing rugby	0.88
two men are talking	0.87
two dogs are playing with each other	0.87

Ranking by Dependency Tree-LSTM model	Score
a woman is slicing potatoes	
a woman is cutting potatoes	4.82
potatoes are being sliced by a woman	4.70
tofu is being sliced by a woman	4.39
a boy is waving at some young runners from the ocean	
a group of men is playing with a ball on the beach	3.79
a young boy wearing a red swimsuit is jumping out of a blue kiddies pool	3.37
the man is tossing a kid into the swimming pool that is near the ocean	3.19
two men are playing guitar	
the man is singing and playing the guitar	4.08
the man is opening the guitar for donations and plays with the case	4.01
two men are dancing and singing in front of a crowd	4.00

Table 4: Most similar sentences from a 1000-sentence sample drawn from the SICK test set. The Tree-LSTM model is able to pick up on more subtle relationships, such as that between "beach" and "ocean" in the second example.

Pennington et al., 2014) have found wide applicability in a variety of NLP tasks. Following this success, there has been substantial interest in the area of learning distributed phrase and sentence representations (Mitchell and Lapata, 2010; Yessenalina and Cardie, 2011; Grefenstette et al., 2013; Mikolov et al., 2013), as well as distributed representations of longer bodies of text such as paragraphs and documents (Srivastava et al., 2013; Le and Mikolov, 2014).

Our approach builds on recursive neural networks (Goller and Kuchler, 1996; Socher et al., 2011), which we abbreviate as Tree-RNNs in order to avoid confusion with recurrent neural networks. Under the Tree-RNN framework, the vector representation associated with each node of a tree is composed as a function of the vectors corresponding to the children of the node. The choice of composition function gives rise to numerous variants of this basic framework. Tree-RNNs have been used to parse images of natural scenes (Socher et al., 2011), compose phrase representations from word vectors (Socher et al., 2012), and classify the sentiment polarity of sentences (Socher et al., 2013).

9 Conclusion

In this paper, we introduced a generalization of LSTMs to tree-structured network topologies. The Tree-LSTM architecture can be applied to trees with arbitrary branching factor. We demonstrated the effectiveness of the Tree-LSTM by applying the architecture in two tasks: semantic relatedness and sentiment classification, outperforming existing systems on both. Controlling for model dimensionality, we demonstrated that Tree-LSTM models are able to outperform their sequential counterparts. Our results suggest further lines of work in characterizing the role of structure in producing distributed representations of sentences.

Acknowledgements

We thank our anonymous reviewers for their valuable feedback. Stanford University gratefully acknowledges the support of a Natural Language Understanding-focused gift from Google Inc. and the Defense Advanced Research Projects Agency (DARPA) Deep Exploration and Filtering of Text (DEFT) Program under Air Force Research Laboratory (AFRL) contract no. FA8750-13-2-0040. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the DARPA, AFRL, or the US government.

References

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*.

Bengio, Yoshua, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks* 5(2).

Bjerva, Johannes, Johan Bos, Rob van der Goot, and Malvina Nissim. 2014. The Meaning Factory: Formal semantics for recognizing textual entailment and determining semantic similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*.

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

Chen, Danqi and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Collobert, Ronan, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research* 12:2493–2537.

Duchi, John, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research* 12:2121–2159.

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

Foltz, Peter W, Walter Kintsch, and Thomas K Landauer. 1998. The measurement of textual coherence with latent semantic analysis. *Discourse processes* 25(2-3):285–307.

Ganitkevitch, Juri, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The Paraphrase Database. In *Proceedings of HLT-NAACL 2013*.

Goller, Christoph and Andreas Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *IEEE International Conference on Neural Networks*.

Graves, Alex, Navdeep Jaitly, and A-R Mohamed. 2013. Hybrid speech recognition with deep bidirectional LSTM. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*.

Grefenstette, Edward, Georgiana Dinu, Yao-Zhong Zhang, Mehrnoosh Sadrzadeh, and Marco Baroni. 2013. Multi-step regression learning for compositional distributional semantics. In *Proceedings of the 10th International Conference on Computational Semantics*.

Hochreiter, Sepp. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6(02):107–116.

Hochreiter, Sepp and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9(8).

Huang, Eric H., Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Irsoy, Ozan and Claire Cardie. 2014. Deep recursive neural networks for compositionality in language. In *Advances in Neural Information Processing Systems*.

Jimenez, Sergio, George Duenas, Julia Baquero, Alexander Gelbukh, Av Juan Dios Bátiz, and Av Mendizábal. 2014. UNAL-NLP: Combining soft cardinality features for semantic textual similarity, relatedness and entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*.

Kim, Yoon. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Klein, Dan and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*.

Lai, Alice and Julia Hockenmaier. 2014. Illinois-LH: A denotational and distributional approach to semantics. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*.

Landauer, Thomas K and Susan T Dumais. 1997. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review* 104(2):211.

Le, Quoc and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In

Proceedings of the 31st International Conference on Machine Learning (ICML-14).

Marelli, Marco, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. SemEval-2014 Task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*.

Mikolov, Tomáš. 2012. *Statistical Language Models Based on Neural Networks*. Ph.D. thesis, Brno University of Technology.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*.

Mitchell, Jeff and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive science* 34(8):1388–1429.

Pennington, Jeffrey, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empiricial Methods in Natural Language Processing (EMNLP)*.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams. 1988. Learning representations by back-propagating errors. *Cognitive modeling* 5.

Socher, Richard, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP)*.

Socher, Richard, Andrej Karpathy, Quoc V Le, Christopher D Manning, and Andrew Y Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics* 2.

Socher, Richard, Cliff C Lin, Chris Manning, and Andrew Y Ng. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*.

Socher, Richard, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15:1929–1958.

Srivastava, Nitish, Ruslan Salakhutdinov, and Geoffrey Hinton. 2013. Modeling documents with a Deep Boltzmann Machine. In *Uncertainty in Artificial Intelligence*.

Sutskever, Ilya, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*.

Turian, Joseph, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*.

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

Yessenalina, Ainur and Claire Cardie. 2011. Compositional matrix-space models for sentiment analysis. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

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

Zhao, Jiang, Tian Tian Zhu, and Man Lan. 2014. ECNU: One stone two birds: Ensemble of heterogenous measures for semantic relatedness and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*.