The Lifted Matrix-Space Model for Semantic Composition

WooJin Chung[1]
woojin@nyu.edu

Sheng-Fu Wang[1]
shengfu.wang@nyu.edu

Samuel R. Bowman[1,2]
bowman@nyu.edu

[1]Dept. of Linguistics
New York University
10 Washington Place
New York, NY 10003

[2]Center for Data Science
New York University
60 Fifth Avenue
New York, NY 10011

Abstract

Tree-structured neural network architectures for sentence encoding draw inspiration from the approach to semantic composition generally seen in formal linguistics, and have shown empirical improvements over comparable sequence models by doing so. Moreover, adding multiplicative interaction terms to the composition functions in these models can yield significant further improvements. However, existing compositional approaches that adopt such a powerful composition function scale poorly, with parameter counts exploding as model dimension or vocabulary size grows. We introduce the Lifted Matrix-Space model, which uses a global transformation to map vector word embeddings to matrices, which can then be composed via an operation based on matrix-matrix multiplication. Its composition function effectively transmits a larger number of activations across layers with relatively few model parameters. We evaluate our model on the Stanford NLI corpus, the MultiGenre NLI corpus, and the Stanford Sentiment Treebank and find that it consistently outperforms TreeLSTM (Tai et al., 2015), the previous best known composition function for tree-structured models.

1 Introduction

Contemporary theoretical accounts of natural language syntax and semantics consistently hold that sentences are tree-structured, and that the meaning of each node in each tree is calculated from the meaning of its child nodes using a relatively simple *semantic composition* process which is applied recursively bottom-up (Chierchia and McConnell-Ginet, 1990; Dowty, 2007). In tree-structured recursive neural networks (TreeRNN; Socher et al., 2010), a similar procedure is used to build representations for sentences for use in natural language understanding tasks, with distributed representations for words repeatedly fed through a neural network *composition function* according to a binary tree structure supplied by a parser. The success of a tree-structured model largely depends on the design of its composition function.

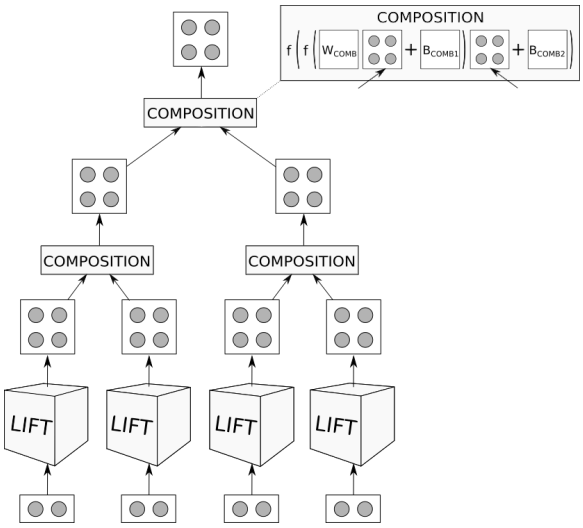


Figure 1: The Lifted Matrix-Space model in schematic form. Words are stored as vectors and projected into matrix space by the LIFT layer. A parametric COMPOSITION function combines pairs of these matrices using multiplicative interactions.

It has been repeatedly shown that a composition function that captures multiplicative interactions between the two items being composed yields better results (Rudolph and Giesbrecht, 2010; Socher et al., 2012, 2013) than do otherwise-equivalent functions based on simple linear interactions. This paper presents a novel model which advances this line of research, the Lifted Matrix-Space model. We utilize a tensor-parameterized LIFT layer that learns to produce matrix representations of words that are dependent on the content of pre-trained word embedding vectors. Composition of two matrix representations is carried out by a composition layer, into which the two matrices are sequentially

Proceedings of the 22nd Conference on Computational Natural Language Learning (CoNLL 2018), pages 508–518
Brussels, Belgium, October 31 - November 1, 2018.

fed. Figure 1 illustrates the model design.

Our model was inspired by Continuation Semantics (Barker and Shan, 2014; Charlow, 2014), where each symbolic representation of words is converted to a higher-order function. There is a consensus in linguistic semantics that a subset of natural language expressions correspond to higher-order functions. Inspired by the works in programming language theory, Continuation Semantics takes a step further and claims that all expressions have to be converted into a higher-order function before they participate in semantic composition. The theory bridges a gap between linguistic semantics and programming language theory, and reinterprets various linguistic phenomena from the view of computation. While we do not directly implement Continuation Semantics, we follow its rough contours: We convert low-level representations (vectors) to higher-order functions (matrices), and composition only takes place between the higher-order functions.

A number of models have been developed to capture the multiplicative interactions between distributed representations. While having a similar objective, the proposed model requires fewer model parameters than the predecessors because it does not necessarily learn each word matrix representation separately, and the number of parameters for the composition function is not proportional to the cube of the hidden state dimension. Because of this, it can be trained with larger vocabularies and more hidden state activations than was possible with its predecessors.

We evaluate our model primarily on the task of *natural language inference* (NLI; MacCartney, 2009). The task consists in determining the inferential relation between a given pair of sentences. It is a principled and widely-used evaluation task for natural language understanding, and knowing the inferential relations is closely related to understanding the meaning of an expression (Chierchia and McConnell-Ginet, 1990). While other tasks such as question answering or machine translation require a model to learn task-specific behavior that goes beyond understanding sentence meaning, NLI results highlight sentence understanding performance in isolation. We also include an evaluation on sentiment classification for comparison with some earlier work.

We find that our model outperforms existing approaches to tree-structured modeling on all three tasks, though it does not set the state of the art on any of them, falling behind other types of complex model. We nonetheless expect that this method will be a valuable ingredient in future models for sentence understanding and a valuable platform for research of compositionality in learned representations.

2 Related work

Composition functions for tree-structured models have been thoroughly studied in recent years (Mitchell and Lapata, 2010; Baroni and Zamparelli, 2010; Zanzotto et al., 2010; Socher et al., 2011). While this line of research has been successful, the majority of the existing models ultimately rely on the additive linear combination of vectors. The Tree-structured recursive neural networks (TreeRNN) of Socher et al. (2010) compose two child node vectors $\vec{h}_l$ and $\vec{h}_r$ using this method:

$$(1) \quad \vec{h} = tanh(W \begin{bmatrix} \vec{h}_l \\ \vec{h}_r \end{bmatrix} + \vec{b})$$

where $\vec{h}_l$, $\vec{h}_r$, $\vec{h}$, $\vec{b} \in \mathbb{R}^{d \times 1}$, and $W \in \mathbb{R}^{d \times 2d}$. Throughout this paper, d stands for the number of activations of a given model.

However, there is no reason to believe that the additive linear combination of vectors is adequate for modeling semantic composition. Formal work in linguistic semantics has shown that many linguistic expressions are well-represented as functions. Accordingly, composing two meanings typically require feeding an argument into a function (function application; Heim and Kratzer, 1998). Such an operation involves a complex interaction between the two meanings, but the classic TreeRNN does not supply any additional means to capture the interaction.

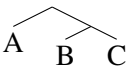
Rudolph and Giesbrecht (2010) report that matrix multiplication, as opposed to element-wise addition, is more suitable for semantic composition. Their Compositional Matrix-Space model (CMS) represents words and phrases as matrices, and they are composed via a simple matrix multiplication:

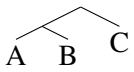
$$(2) \quad P = AB$$

where $A, B, P \in \mathbb{R}^{\mathrm{d} \times \mathrm{d}}$ are matrix representations of the word embeddings. They provide a formal proof that element-wise addition/multiplication of vectors can be subsumed under matrix multiplication. Moreover, they claim that the order-sensitivity of matrix multiplication is adequate for

capturing the semantic composition because natural language is order-sensitive.

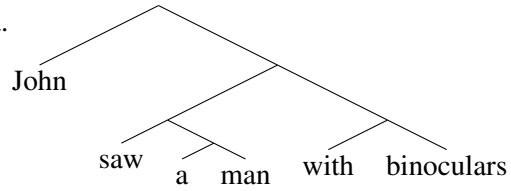
However, as Socher et al. (2012) note, CMS loses syntactic information during composition due to the associative character of matrix multiplication. For instance, the following two tree structures in (3) are syntactically distinct, but CMS would produce the same result for both structures because its mode of composition is associative.

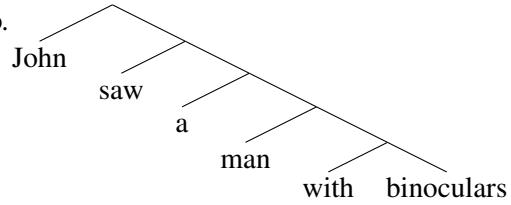
(3) a. 

b. 

CMS cannot distinguish the meaning of the two tree structures and invariably produces ABC (a sequence of matrix multiplications). Therefore, the information on syntactic constituency would be lost. This makes the model less desirable for handling semantic composition of natural language expressions for two reasons: First, the principle of compositionality is violated. Much of the success of the tree-structured models can be credited to the shared hypothesis that the meaning of every tree node is derived from the meanings of its child nodes. Abandoning this principle of compositionality gives up the advantage. Second, it cannot handle structural ambiguities exemplified in (4).

(4) John saw a man with binoculars.

(5) a. 

b. 

The sentence has two interpretations that can be disambiguated with the following paraphrases: (i) John saw a man via binoculars, and (ii) John saw a man who has binoculars. The common syntactic analysis of the ambiguity is that the prepositional phrase *with binoculars* can attach to two different locations. If it attaches to the verb phrase *saw a man*, the first interpretation arises. On the other hand, if it attaches to the noun *man*, the second interpretation is given. However, if the structural information is lost, we would have no way to disambiguate the two readings.

Socher et al.'s (2012) Matrix-Vector RNN (MV-RNN) is another attempt to capture the multiplicative interactions between two vectors, while conforming to the principle of compositionality. They hypothesize that representing operators as matrices can better reflect operator semantics. For each lexical item, a matrix (trained parameter) is assigned in addition to the pre-trained word embedding vector. The model aims to assign the right matrix representations to operators while assigning an identity matrix to words with no operator meaning. One step of semantic composition is defined as follows:

$$\vec{h} = f(B\vec{a}, A\vec{b}) = g(W \begin{bmatrix} B\vec{a} \\ A\vec{b} \end{bmatrix}) \quad (6)$$

$$H = f_M(A, B) = W_M \begin{bmatrix} A \\ B \end{bmatrix} \quad (7)$$

where $\vec{a}, \vec{b}, \vec{h} \in \mathbb{R}^{d \times 1}$, $A, B, H \in \mathbb{R}^{d \times d}$, and W, $W_M \in \mathbb{R}^{d \times 2d}$.

MV-RNN is computationally costly as it needs to learn an additional $d \times d$ matrix for each lexical item. It is empirically known that the size of the vocabulary is roughly proportional to the size of the corpus (Heaps' law; Herdan, 1960), therefore the number of model parameters increases as the corpus gets bigger. This makes the model less ideal for handling a large corpus: having a huge number of parameters causes a problem both for memory usage and for learning efficiency.

Chen et al. (2013) and Socher et al. (2013) present the recursive neural tensor network (RNTN) which reduces the computational complexity of MV-RNN, while capturing the multiplicative interactions between child vectors. The model introduces a third-order tensor $\boldsymbol{V}$ which interacts with the child node vectors as follows:

$$\vec{h} = tanh(W \begin{bmatrix} \vec{h}_l \\ \vec{h}_r \end{bmatrix} + \vec{b} + \vec{h}_l^T \mathbf{V} \vec{h}_r) \quad (8)$$

where $\vec{h}_l$, $\vec{h}_r$, $\vec{b} \in \mathbb{R}^{d \times 1}$, $\mathbf{W} \in \mathbb{R}^{d \times 2d}$, and $\mathbf{V} \in \mathbb{R}^{d \times d \times d}$. RNTN improves on MV-RNN in that its parameter size is not proportional to the size of the corpus. However, the addition of the third-order tensor $\mathbf{V}$ of dimension $d \times d \times d$ still requires proportionally more parameters.

The last composition function relevant to this paper is the Tree-structured long short-term memory networks (TreeLSTM; Tai et al., 2015; Zhu et al., 2015; Le and Zuidema, 2015), particularly the version over a constituency tree. It is an

Model	Params.	Associative	Multiplicative	Activation size w.r.t. TreeRNN
TreeRNN/LSTM	O($d \times d$)	No	No	1
CMS	O($V \times d \times d$)	Yes	Yes	$1/V$
MV-RNN	O($V \times d \times d$)	No	Yes	$1/V$
RNTN	O($d \times d \times d$)	No	Yes	$1/d$
LMS (this work)	O($d \times d_{emb}$)	No	Yes	$1/d_{emb}$

Table 1: Summary of the models. *Params.* is the number of model parameters (not counting pretrained word vectors), d is the number of activations at each tree node, d_{emb} is the dimension of the word embeddings, and V is the size of the vocabulary. *Associative* and *Multiplicative* indicate whether composition is associative and whether it includes multiplicative interactions between inputs, respectively. *Activation size w.r.t. TreeRNN* shows how activation sizes scale with respect to TreeRNN when all of the models have the same parameter count.

extension of TreeRNN which adapts long short-term memory (LSTM; Hochreiter and Schmidhuber, 1997) networks. It shares the advantage of LSTM networks in that it prevents the vanishing gradient problem (Hochreiter et al., 2001).

Unlike TreeRNN, the output hidden state $\vec{h}$ of TreeLSTM is not directly calculated from the hidden states of its child nodes, $\vec{h}_l$ and $\vec{h}_r$. Rather, each node in TreeLSTM maintains a cell state $\vec{c}$ that keeps track of important information of its child nodes. The output hidden state $\vec{h}$ is drawn from the cell state $\vec{c}$ by passing it through an output gate $\vec{o}$.

The cell state is calculated in three steps: (i) Compute a new candidate $\vec{g}$ from $\vec{h}_l$ and $\vec{h}_r$. TreeLSTM selects which values to take from the new candidate $\vec{g}$ by passing it through an input gate $\vec{i}$. (ii) Choose which values to forget from the cell states of the child nodes, $\vec{c}_l$ and $\vec{c}_r$. For each child node, an element-wise product ($\odot$) between its cell state and the forget gate (either $\vec{f}_l$ and $\vec{f}_r$, depending on the child node) is calculated. (iii) Lastly, sum up the results from (i) and (ii).

$$\vec{g} = tanh\left(W \begin{bmatrix} \vec{h}_l \\ \vec{h}_r \end{bmatrix} + \vec{b}\right) \tag{9}$$

$$\begin{bmatrix} \vec{i} \\ \vec{f}_l \\ \vec{f}_r \\ \vec{o} \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \sigma \end{bmatrix} \left(W \begin{bmatrix} \vec{h}_l \\ \vec{h}_r \end{bmatrix} + \vec{b}\right) \tag{10}$$

$$\vec{c} = \vec{f}_l \odot \vec{c}_l + \vec{f}_r \odot \vec{c}_r + \vec{i} \odot \vec{g} \tag{11}$$

$$\vec{h} = \vec{o} \odot tanh(\vec{c}) \tag{12}$$

TreeLSTM achieves the state-of-the-art performance among the tree-structured models in various tasks, including natural language inference and sentiment classification. However, there are non-tree-structured models on the market that outperform TreeLSTM. Our goal is to design a stronger composition function that enhances the performance of tree-structured models. We develop a composition function that captures the multiplicative interaction between distributed representations. At the same time, we improve on the predecessors in terms of scalability, making the model more suitable for larger datasets.

To recapitulate, TreeRNN and TreeLSTM reflect the principle of compositionality but cannot capture the multiplicative interaction between two expressions. In contrast, CMS incorporates multiplicative interaction but violates the principle of compositionality. MV-RNN is compositional and also captures the multiplicative interaction, but it requires a learned $d \times d$ matrix for each vocabulary item. RNTN is also compositional and incorporates multiplicative interaction, but it requires less parameters than MV-RNN. Nevertheless, it requires significantly more parameters than TreeRNN or TreeLSTM. Table 1 is an overview of the discussed models.

Other interesting works enrich semantic composition with additional context such as grammatical roles or function/argumenthood (Clark et al., 2008; Erk and Padó, 2008; Grefenstette et al., 2014; Asher et al., 2016; Weir et al., 2016).

3 The Lifted Matrix-Space model

3.1 Base model

We present the Lifted Matrix-Space model (LMS) which renders semantic composition in a novel way. Our model consists of three subparts: the LIFT layer, the composition layer, and the TreeLSTM wrapper. The LIFT layer takes a word embedding vector and outputs a corresponding $\sqrt{d} \times \sqrt{d}$ matrix (eq. 13).

$$H = tanh(\mathbf{W}_{\text{LIFT}}\vec{c} + B_{\text{LIFT}}) \tag{13}$$

where $\vec{c} \in \mathbb{R}^{d_{emb} \times 1}$, $B_{\text{LIFT}} \in \mathbb{R}^{\sqrt{d} \times \sqrt{d}}$, and $\mathbf{W}_{\text{LIFT}} \in \mathbb{R}^{\sqrt{d} \times \sqrt{d} \times d_{emb}}$. The resulting H matrix serves as an input for the composition layer.

Given the matrix representations of two child nodes, H_l and H_r, the composition layer first takes H_l and returns a hidden state $H_{inner} \in \mathbb{R}^{\sqrt{d} \times \sqrt{d}}$ (eq. 14). Since H_{inner} is also a matrix, it can function as the weight matrix for H_r. The composition layer multiplies H_{inner} with H_r, adds a bias, and feeds the result into a non-linear activation function (eq. 15). This yields $H_{cand} \in \mathbb{R}^{\sqrt{d} \times \sqrt{d}}$, which for the base model is the output of semantic composition.

$$H_{inner} = tanh(W_{\text{COMB}} H_l + B_{\text{COMB1}}) \quad (14)$$

$$H_{cand} = tanh(H_{inner} H_r + B_{\text{COMB2}}) \quad (15)$$

As in CMS, the primary mode of semantic composition is matrix multiplication. However, LMS improves on CMS in that it avoids associativity. LMS differs from MV-RNN in that it does not learn a $d \times d$ matrix for each vocabulary item. Compared to RNTN, LMS transmits a larger number of activations across layers, given the same parameter count. In both models, the size of the third-order tensor is the dominant factor in determining the number of model parameters. The parameter count of LMS is approximately proportional to the number of activations (d), but the parameter count of RNTN is approximately proportional to the cube of the number of activations (d^3). Therefore, LMS can transmit the same number of activations with fewer model parameters.

3.2 LMS augmented with LSTM components

We augment the base model with LSTM components (LMS-LSTM) to circumvent the problem of long-term dependencies. As in the case of TreeLSTM, we additionally manage cell states ($\vec{c}_l$, $\vec{c}_r$). Since the LSTM components operate on vectors, we reshape H_{cand}, H_l, and H_r into $d \times 1$ column vectors respectively, and produce $\vec{g}$, $\vec{h}_l$, and $\vec{h}_r$. The output of the LSTM components are calculated based on these vectors, and is reshaped back to a $\sqrt{d} \times \sqrt{d}$ matrix (eq. 22).

$$\vec{g} = \text{VECTORIZE}(H_{cand}) \quad (16)$$

$$\vec{h}_l = \text{VECTORIZE}(H_l) \quad (17)$$

$$\vec{h}_r = \text{VECTORIZE}(H_r) \quad (18)$$

$$\begin{bmatrix} \vec{i} \\ \vec{f}_l \\ \vec{f}_r \\ \vec{o} \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \sigma \end{bmatrix} \left(W \begin{bmatrix} \vec{h}_l \\ \vec{h}_r \end{bmatrix} + B \right) \quad (19)$$

$$\vec{c} = \vec{f}_l \odot \vec{c}_l + \vec{f}_r \odot \vec{c}_r + \vec{i} \odot \vec{g} \quad (20)$$

$$\vec{h} = \vec{o} \odot tanh(\vec{c}) \quad (21)$$

$$H = \text{TO-MATRIX}(\vec{h}) \quad (22)$$

3.3 Simplified variants

We implement two LMS-LSTM variants with a simpler composition function as an ablation study. The first variant replaces the equations in (14) and (15) with a single equation (eq. 23), which does not utilize a weight matrix. It simply multiplies the matrix representations of two child nodes $H_l, H_r \in \mathbb{R}^{\sqrt{d} \times \sqrt{d}}$, adds a bias $B_{\text{COMB}} \in \mathbb{R}^{\sqrt{d} \times \sqrt{d}}$, and feeds the result into a non-linear activation function.

$$H_{cand} = tanh(H_l H_r + B_{\text{COMB}}) \quad (23)$$

The second variant is more complex than the first, in a way that a weight matrix $W_{\text{COMB}} \in \mathbb{R}^{\sqrt{d} \times \sqrt{d}}$ is added to the equation (eq. 24). But unlike the full LMS-LSTM which has two *tanh* layers, it only utilizes one.

$$H_{cand} = tanh(W_{\text{COMB}} H_l H_r + B_{\text{COMB}}) \quad (24)$$

4 Experiments

4.1 Implementation details

As our interest is in the performance of composition functions, we compare LMS-LSTM with TreeLSTM, the previous best known composition function for tree-structured models. To allow for efficient batching, we use the SPINN-PI-NT approach (Bowman et al., 2016), which implements standard TreeLSTM using stack and buffer data structures borrowed from parsing, rather than tree structures. We implement our model by replacing SPINN-PI-NT's composition function with ours and adding the LIFT layer.

We use the 300D reference GloVe vectors (840B token version; Pennington et al., 2014) for word representations. We fine-tune the word embeddings for improved results. We follow Bowman et al. (2016) and other prior work in our use of an MLP with product and difference features to classify pairs of sentences.

$$(25) \quad \vec{x}_{classifier} = \begin{bmatrix} \vec{h}_{premise} \\ \vec{h}_{hypothesis} \\ \vec{h}_{premise} - \vec{h}_{hypothesis} \\ \vec{h}_{premise} \odot \vec{h}_{hypothesis} \end{bmatrix}$$

The feature vector is fed into an MLP that consists of two ReLU neural network layers and a softmax layer. In both models, the objective function is a sum of a cross-entropy loss function and an L2 regularization term. Both models use the Adam optimizer (Kingma and Ba, 2014). Dropout (Srivastava et al., 2014) is applied to the classifier and to the word embeddings. The MLP layer also utilizes Layer Normalization (Ba et al., 2016).[1]

4.2 Datasets

We first train and test our models on the *Stanford Natural Language Inference corpus* (SNLI; Bowman et al., 2015). The SNLI corpus contains 570,152 pairs of natural language sentences that are labeled for *entailment*, *contradiction*, and *neutral*. It consists of sentences that were written and validated by humans. Along with the MultiNLI corpus introduced below, it is two orders of magnitude larger than other human-authored resources for NLI tasks. The following example illustrates the general format of the corpus.

(26) **PREMISE:** A soccer game with multiple males playing.
HYPOTHESIS: Some men are playing a sport.
LABEL: Entailment

We test our models on the *Multi-Genre Natural Language Inference corpus* (MultiNLI; Williams et al., 2017). The corpus consists of 433k pairs of examples, and each pair is labeled for *entailment*, *contradiction*, and *neutral*. MultiNLI has the same format as SNLI, so it is possible to train on both datasets at the same time (as we do when testing on MultiNLI). Two notable features distinguish MultiNLI from SNLI: (i) It is collected from ten distinct genres of spoken and written English. This makes the dataset more representative of human language use. (ii) The examples in MultiNLI are considerably longer than the ones in SNLI. These two features make MultiNLI classification fairly more difficult than SNLI. The pair of sentences in (27) is an illustrative example. The sentences are from the section of the corpus that is transcribed verbatim from telephone speech.

(27) **GENRE:** Telephone speech
PREMISE: Yes now you know if if everybody like in August when everybody's on vacation or something we can dress a little more casual or
HYPOTHESIS: August is a black out month for vacations in the company.
LABEL: Contradiction

The MultiNLI training set consists of five different genres of spoken and written English, the *matched* test set contains sentence pairs from only those five genres, and the *mismatched* test set contains sentence pairs from additional genres.

We also experiment on the Stanford Sentiment Treebank (SST; Socher et al., 2013), which is constructed by extracting movie review excerpts written in English from `rottentomatoes.com`, and labeling them with Amazon Mechanical Turk. Each example in SST is paired with a parse tree, and each node of the tree is tagged with a fine-grained sentiment label (5 classes).

5 Results and Analysis

Table 2 summarizes the results on SNLI and MultiNLI classification. We use the same preprocessing steps for all results we report, including loading the parse trees supplied with the datasets. Dropout rate, size of activations, number and size of MLP layers, and L2 regularization term are tuned using repeated random search. MV-RNN and RNTN introduced in the earlier sections are extremely expensive in terms of computational resources, and training the models with comparative hyperparameter settings quickly runs out of memory on a high-end GPU. We do not include them in the comparison for this reason. TreeLSTM performs the best with one MLP layer, while LMS-LSTM displays the best performance with two MLP layers. The difference in parameter count is largely affected by this choice, and in principle one model does not demand notably more computational resources than the other.

On the SNLI test set, LMS-LSTM has an additional 1.3% gain over TreeLSTM. Also, both of the simplified variants of LMS-LSTM outperform TreeLSTM, but by a smaller margin. On the MultiNLI test sets, LMS-LSTM scores 1.3% higher on the matched test set and 1.9% higher on the mismatched test set.

[1] The source code and the checkpoints for the models trained for the NLI tasks are available at `https://github.com/nyu-mll/spinn`.

Model	Params.	S tr.	S te.	M tr.	M te. mat.	M te. mism.
Baselines						
CBOW (Williams et al., 2017)	–	–	80.6	–	65.2	64.6
BiLSTM (Williams et al., 2017)	2.8m	–	81.5	–	67.5	67.1
Shortcut-Stacked BiLSTM (Nie and Bansal, 2017)	34.7m	–	86.1	–	74.6	73.6
DIIN (Gong et al., 2018)	–	–	**88.0**	–	**78.8**	**77.8**
Existing Tree-Structured Model Runs						
300D TreeLSTM (Bowman et al., 2016)	3.4m	84.4	80.9	–	–	–
300D SPINN-PI (Bowman et al., 2016)	3.7m	89.2	83.2	–	–	–
Our Experiments						
441D LMS (base)	2.0(+11.6m)	79.7	76.5	–	–	–
441D LMS-LSTM (simplified, $-W_{\text{COMB}}$, $-tanh$)	3.3(+11.6)m	90.5	84.1	–	–	–
324D LMS-LSTM (simplified, $-tanh$)	2.2(+11.6)m	92.5	84.5	–	–	–
700D TreeLSTM	2.0(+11.6)m	89.5	83.6	–	–	–
576D LMS-LSTM (full)	4.6(+11.6)m	86.0	84.9	–	–	–
700D TreeLSTM	4.6(+30.2)m	–	–	78.9	70.0	69.7
576D LMS-LSTM (full)	5.9(+30.2)m	–	–	80.5	71.3	71.6

Table 2: Results on NLI classification with sentence-to-vector encoders. *Params.* is the approximate number of model parameters, and the numbers in parentheses indicate the parameters contributed by word embeddings. *S tr.*, and *S te.* are the class accuracies (%) on SNLI train set and test set, respectively. *M tr.*, *M te. mat.*, and *M te. mism.* are the class accuracies (%) on MultiNLI train set, matched test set, and mismatched test set, respectively. Underlining marks the best result among tree-structured models.

We cite the state-of-the-art results of non-tree-structured models, although these models are only relevant for our absolute performance numbers. The Shortcut-Stacked sentence encoder achieves the state-of-the-art result among non-attention-based models, outperforming LMS-LSTM. While this paper focuses on the design of the composition function, we expect that adding depth along the lines of Irsoy and Cardie (2014) and shortcut connections to LMS-LSTM would offer comparable results. Gong et al.'s (2018) attention-based Densely Interactive Inference Network (DIIN) displays the state-of-the-art performance among all models. Applying various attention mechanisms to tree-structured models is left for future research.

We inspect the performance of the models on certain subsets of the MultiNLI corpus that manifest linguistically difficult phenomena, which was categorized by Williams et al. (2017). The phenomena include pronouns, quantifiers (e.g., *every*, *each*, *some*), modals (e.g., *must*, *can*), negation, wh-terms (e.g., *who*, *where*), belief verbs (e.g., *believe*, *think*), time terms (e.g., *then*, *today*), discourse markers (e.g., *but*, *thus*), presupposition triggers (e.g., *again*, *too*), and so on. In linguistic semantics, these phenomena are known to involve complex interactions that are more intricate than a simple merger of concepts. For instance, modals express possibilities or necessities that are beyond "here and now". One can say 'John might be home' to convey that there is a possibility that John is home. The utterance is perfectly compatible with a situation in which John is in fact at school, so modals like *might* let us reason about things that are potentially false in the real world. We use the same code as Williams et al. (2017) to

	LMS-LSTM		TreeLSTM	
Phenomenon	**Mat.**	**Mismat.**	**Mat.**	**Mismat.**
Pronoun	**72.0**	**71.6**	69.6	70.3
Quantifier	**72.2**	**71.7**	69.9	70.5
Modal	**70.6**	**70.8**	69.8	69.2
Negation	**72.3**	**74.1**	70.3	72.4
Wh-term	**70.5**	**69.7**	68.6	68.6
Belief verb	**70.1**	**70.1**	68.4	68.7
Time term	**70.0**	**71.1**	67.3	69.4
Discourse mark.	**68.8**	**68.8**	67.0	67.0
Presup. triggers	**71.5**	**71.9**	69.1	69.9
Compr./Supr.	**69.0**	**67.5**	67.0	67.1
Conditionals	**69.7**	**71.3**	68.2	70.5
Tense match	**73.3**	**72.5**	71.0	71.2
Interjection	**69.7**	**74.3**	**69.7**	72.5
Adj/Adv	**72.6**	**72.0**	70.3	70.7
Determiner	**72.4**	**72.1**	70.3	70.8
Length 0-10	**72.8**	**72.8**	69.8	72.7
Length 11-14	**72.6**	**72.8**	72	70.5
Length 15-19	**71.0**	**70.8**	69.3	68.3
Length 20+	**75.2**	68.2	69.8	**69.0**

Table 3: MultiNLI development set classification accuracies (%), classified using the tags introduced in Williams et al. (2017).

Model	Test
Baselines	
MV-RNN (Socher et al., 2013)	44.4
RNTN (Socher et al., 2013)	45.7
Deep RNN (Irsoy and Cardie, 2014)	49.8
TreeLSTM (Tai et al., 2015)	51.0
TreeBiGRU w/ attention (Kokkinos and Potamianos, 2017)	52.4
Our Experiments	
312D TreeLSTM	48.9
144D LMS-LSTM	50.1

Table 4: Five-way test set classification accuracies (%) on the Stanford Sentiment Treebank.

Category	# of samples	Ratio (%)
NP	63346	43.31
VP	30534	20.08
PP	25624	17.98
ROOT	18267	12.49
S	4863	3.06
SBAR	2004	1.20
ADVP	1136	0.27
ADJP	408	0.13
Etc.	160	1.45

Table 5: Syntactic category distribution of SNLI development set, classified using the tags introduced in Bowman et al. (2015).

categorize the data to make fair comparisons.

In addition to the categories offered by Williams et al. (2017), we inspect whether sentences containing adjectives/adverbs affect the performance of the models. Baroni and Zamparelli (2010) show that adjectives are better represented as matrices, as opposed to vectors. We also inspect whether the presence of a determiner in the hypothesis that refers back to a salient referent in the premise affects the model performance. Determiners are known to encode intricate properties in linguistic semantics and have been one of the major research topics (Elbourne, 2005; Charlow, 2014). Lastly, we examine whether the performance of the models varies with respect to sentence length, as longer sentences are harder to comprehend.

Table 3 summarizes the result of the inspection on linguistically difficult phenomena. We see gains uniformly across the board, but with particularly clear gains on negation (+2% on the matched set/+1.7% on the mismatched set), quantifiers (+2.3%/+1.2%), time terms (+2.7%/+1.7%), tense matches (+2.3%/+1.3%), adjectives/adverbs (+2.3%/+1.3%), and longer sentences (length 15-19: +1.7%/+2.5%; length 20+: +5.4%/-0.8%).

Table 4 summarizes the results on SST classification, particularly on the fine-grained task with 5 classes. While our implementation does not exactly reproduce Tai et al.'s (2015) TreeLSTM results, a comparison between our trained TreeLSTM and LMS-LSTM is consistent with the patterns seen in NLI tasks.

We examine how well the constituent representations produced by LMS-LSTM and TreeLSTM encode syntactic category information. As mentioned earlier, there is a consensus in linguistic semantics that semantic composition involves function application (i.e., feeding an argument to a function) which goes beyond a simple merger of two concepts. Given that the syntactic category of a node determines whether the node serves as a function or an argument in semantic composition, we hypothesize that the distributed representation of each node would encode syntactic category information if the models learned how to do function application. To assess the quality of the representations, we first split the SNLI development set into training and test sets. From the training set, we extract the hidden state of every constituent (i.e., phrase) produced by the best performing models. For each of the models, we train linear classifiers that learn to do the following two tasks: (i) 3-way classification, which trains and tests exclusively on noun phrases, verb phrases, and prepositional phrases, and (ii) 19-way classification, which trains and tests on all 19 category labels attested in the SNLI development set. The distribution of the 19 category labels is provided in Table 5. We opt for a linear classifier to keep the classification process simple, so that we can properly assess the quality of the constituent representations.

Table 6 summarizes the results on the syntactic category classification task. As a baseline, we train a bag-of-words (BOW) model which produces the hidden state of a given phrase by summing the GloVe embeddings of the words of the phrase. We train and test on the hidden states produced by BOW as well. The hidden state representations produced by LMS-LSTM yield the best results on both 3-way and 19-way classification tasks. Comparing LMS-LSTM and TreeLSTM representations, we see a 5.1% gain on the 3-way classification and a 5.5% gain on the 19-way classification.

Model	3-way Train	3-way Test	19-way Train	19-way Test
300D BOW	86.4	85.6	82.7	82.1
700D TreeLSTM	93.2	91.2	90.0	86.6
576D LMS-LSTM	97.3	**96.3**	94.0	**92.1**

Table 6: Syntactic category classification accuracies (%) on SNLI development set, classified using the tags introduced in Bowman et al. (2015).

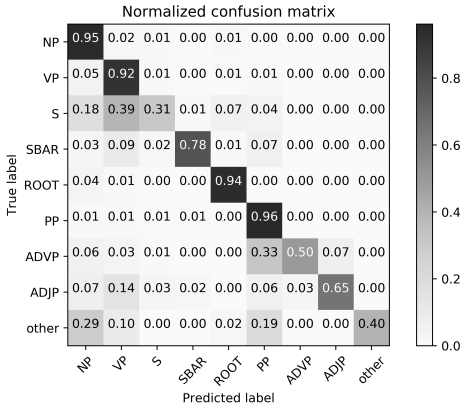
Figure 2 depicts the corresponding confusion matrices for the 19-way classification task. We show the most frequent eight classes due to space limitations. We observe notable gains on adverbial phrases (ADVP; +24%), adjectival phrases (ADJP; +12%), verb phrases (VP; +9%), and clauses introduced by subordinate conjunction (SBAR; +9%). We also observe a considerable gain on non-terminal declarative clauses (S; +9%), although the absolute number is fairly low compared to other categories. While we do not have a full comprehension of the drop in classification accuracy, we speculate that the ambiguity of infinitival clauses and gerund phrases is one of the culprits. As exemplified in (28) and (29) respectively, infinitival clauses and gerund phrases are not only labeled as a VP but also as an S in the SNLI dataset. Given that our experiment is set up in a way that each constituent is assigned exactly one category label, a good number of infinitival clauses and gerund phrases that are labeled as an S could have been classified as a VP, resulting in a drop in classification accuracy. On the other hand, VP constituents are less affected by the ambiguity because the majority of them are neither an infinitival clause nor a gerund phrase, as shown in (30).

(28) A dog carries a snowball [$_{S}$ [$_{VP}$ to give it to his owner]]

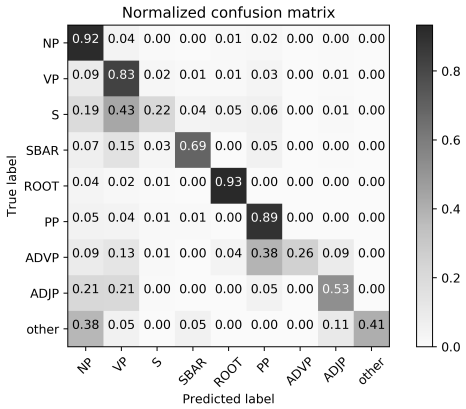
(29) Two women are embracing while [$_{S}$ [$_{VP}$ holding to-go packages after just eating lunch]]

(30) A few people [$_{VP}$ are [$_{VP}$ catching fish]]

With the exclusion of non-terminal declarative clauses, the categories we see notable gains are known to play the role of a function in semantic composition. On the other hand, both models are efficient in identifying noun phrases (NP), which are typically arguments of a function in semantic composition. We speculate that the results are indicative of LMS-LSTM's ability to identify functions and arguments, and this hints that the model is learning to do function application.



(a) LMS-LSTM



(b) TreeLSTM

Figure 2: Confusion matrices of LMS-LSTM and TreeLSTM for 19-way linear classification.

6 Conclusion

In this paper, we propose a novel model for semantic composition that utilizes matrix multiplication. Experimental results indicate that, while our model does not reach the state of the art on any of the three datasets under study, it does substantially outperform all known tree-structured models, and lays a strong foundation for future work on tree-structured compositionality in artificial neural networks.

Acknowledgments

We thank NVIDIA Corporation for the donation of the Titan X Pascal GPU used for this research. This project has benefited from financial support to SB by Google, Tencent Holdings, and Samsung Research.

References

Nicholas Asher, Tim Van de Cruys, Antoine Bride, and Márta Abrusán. 2016. Integrating type theory and distributional semantics: a case study on adjective–noun compositions. *Computational Linguistics*, 42(4):703–725.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer Normalization. *arXiv:1607.06450*.

Chris Barker and Chung-chieh Shan. 2014. *Continuations and natural language*, volume 53. Oxford studies in theoretical linguistics.

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1183–1193.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning, and Christopher Potts. 2016. A fast unified model for parsing and sentence understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Simon Charlow. 2014. *On the semantics of exceptional scope*. Ph.D. thesis, New York University.

Danqi Chen, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2013. Learning new facts from knowledge bases with neural tensor networks and semantic word vectors. In *Proceedings of workshop at ICLR*, pages 1–4.

Gennaro Chierchia and Sally McConnell-Ginet. 1990. *Meaning and grammar: An introduction to semantics*. MIT Press, Cambridge, MA.

Stephen Clark, Bob Coecke, and Mehrnoosh Sadrzadeh. 2008. A compositional distributional model of meaning. In *Proceedings of the Second Quantum Interaction Symposium (QI-2008)*, pages 133–140.

David Dowty. 2007. Compositionality as an empirical problem. In *Direct Compositionality*. Oxford University Press.

Paul D. Elbourne. 2005. *Situations and individuals*, volume 90. Mit Press Cambridge, MA.

Katrin Erk and Sebastian Padó. 2008. A structured vector space model for word meaning in context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 897–906.

Yichen Gong, Heng Luo, and Jian Zhang. 2018. Natural language inference over interaction space. *ICLR 2018*.

Edward Grefenstette, Mehrnoosh Sadrzadeh, Stephen Clark, Bob Coecke, and Stephen Pulman. 2014. Concrete sentence spaces for compositional distributional models of meaning. In *Computing meaning*, pages 71–86. Springer.

Irene Heim and Angelika Kratzer. 1998. *Semantics in Generative Grammar*. Blackwell Publishers.

Gustav Herdan. 1960. *Type-token mathematics*. Mouton.

Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. 2001. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. *A Field Guide to Dynamical Recurrent Networks*, pages 237–243.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1–32.

Ozan Irsoy and Claire Cardie. 2014. Deep recursive neural networks for compositionality in language. In *Proceedings of Advances in Neural Information Processing Systems*, pages 2096–2104.

Diederik Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *The International Conference on Learning Representations*, pages 1–13.

Filippos Kokkinos and Alexandros Potamianos. 2017. Structural Attention Neural Networks for improved sentiment analysis. *arXiv:1701.01811*.

Phong Le and Willem Zuidema. 2015. Compositional Distributional Semantics with Long Short Term Memory. *arXiv:1503.02510*.

Bill MacCartney. 2009. *Natural language inference*. Ph.D. thesis, Stanford University.

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

Yixin Nie and Mohit Bansal. 2017. Shortcut-stacked sentence encoders for multi-domain inference. In *Proceedings of the 2nd Workshop on Evaluating Vector Space Representations for NLP*, pages 41–45.

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

Sebastian Rudolph and Eugenie Giesbrecht. 2010. Compositional matrix-space models of language. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 907–916.

Richard Socher, 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*, pages 1201–1211.

Richard Socher, Cliff C. Lin, Christopher D. 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)*, pages 129–136.

Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*, pages 1–9.

Richard Socher, Alex Perelygin, Jean 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)*, pages 1631–1642.

Nitish Srivastava, Geoffrey E. 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(1):1929–1958.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of ACL*, pages 1556–1566.

David Weir, Julie Weeds, Jeremy Reffin, and Thomas Kober. 2016. Aligning packed dependency trees: a theory of composition for distributional semantics. *Computational Linguistics*, 42(4):727–761.

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2017. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. *arXiv:1704.05426*.

Fabio Massimo Zanzotto, Ioannis Korkontzelos, Francesca Fallucchi, and Suresh Manandhar. 2010. Estimating linear models for compositional distributional semantics. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1263–1271.

Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long Short-Term Memory Over Tree Structures. In *Proc. of ICML*, pages 1604–1612.