# Semantic Compositionality through Recursive Matrix-Vector Spaces

**Richard Socher  Brody Huval  Christopher D. Manning  Andrew Y. Ng**
richard@socher.org, {brodyh,manning,ang}@stanford.edu
Computer Science Department, Stanford University

## Abstract

Single-word vector space models have been very successful at learning lexical information. However, they cannot capture the compositional meaning of longer phrases, preventing them from a deeper understanding of language. We introduce a recursive neural network (RNN) model that learns compositional vector representations for phrases and sentences of arbitrary syntactic type and length. Our model assigns a vector and a matrix to every node in a parse tree: the vector captures the inherent meaning of the constituent, while the matrix captures how it changes the meaning of neighboring words or phrases. This matrix-vector RNN can learn the meaning of operators in propositional logic and natural language. The model obtains state of the art performance on three different experiments: predicting fine-grained sentiment distributions of adverb-adjective pairs; classifying sentiment labels of movie reviews and classifying semantic relationships such as cause-effect or topic-message between nouns using the syntactic path between them.

## 1 Introduction

Semantic word vector spaces are at the core of many useful natural language applications such as search query expansions (Jones et al., 2006), fact extraction for information retrieval (Paşca et al., 2006) and automatic annotation of text with disambiguated Wikipedia links (Ratinov et al., 2011), among many others (Turney and Pantel, 2010). In these models the meaning of a word is encoded as a vector computed from co-occurrence statistics of a word and its neighboring words. Such vectors have been shown to correlate well with human judgments of word similarity (Griffiths et al., 2007).

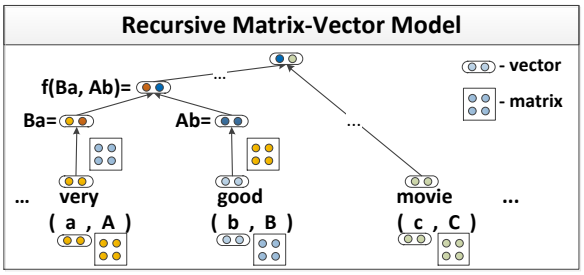


Figure 1: A recursive neural network which learns semantic vector representations of phrases in a tree structure. Each word and phrase is represented by a vector and a matrix, e.g., *very* = $(a, A)$. The matrix is applied to neighboring vectors. The same function is repeated to combine the phrase *very good* with *movie*.

Despite their success, single word vector models are severely limited since they do not capture *compositionality*, the important quality of natural language that allows speakers to determine the meaning of a longer expression based on the meanings of its words and the rules used to combine them (Frege, 1892). This prevents them from gaining a deeper understanding of the semantics of longer phrases or sentences. Recently, there has been much progress in capturing compositionality in vector spaces, e.g., (Mitchell and Lapata, 2010; Baroni and Zamparelli, 2010; Zanzotto et al., 2010; Yessenalina and Cardie, 2011; Socher et al., 2011c) (see related work). We extend these approaches with a more general and powerful model of semantic composition.

We present a novel recursive neural network model for semantic compositionality. In our context, compositionality is the ability to learn compositional vector representations for various types of phrases and sentences of arbitrary length. Fig. 1 shows an illustration of the model in which each constituent (a word or longer phrase) has a matrix-vector (MV)



*Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211, Jeju Island, Korea, 12–14 July 2012. 

representation. The vector captures the meaning of that constituent. The matrix captures how it modifies the meaning of the other word that it combines with. A representation for a longer phrase is computed bottom-up by recursively combining the words according to the syntactic structure of a parse tree. Since the model uses the MV representation with a neural network as the final merging function, we call our model a matrix-vector recursive neural network (MV-RNN).

We show that the ability to capture semantic compositionality in a syntactically plausible way translates into state of the art performance on various tasks. The first experiment demonstrates that our model can learn fine-grained semantic compositionality. The task is to predict a sentiment distribution over movie reviews of adverb-adjective pairs such as *unbelievably sad* or *really awesome*. The MV-RNN is the only model that is able to properly negate sentiment when adjectives are combined with *not*. The MV-RNN outperforms previous state of the art models on full sentence sentiment prediction of movie reviews. The last experiment shows that the MV-RNN can also be used to find relationships between words using the learned phrase vectors. The relationship between words is recursively constructed and composed by words of arbitrary type in the variable length syntactic path between them. On the associated task of classifying relationships between nouns in arbitrary positions of a sentence the model outperforms all previous approaches on the SemEval-2010 Task 8 competition (Hendrickx et al., 2010). It outperforms all but one of the previous approaches without using any hand-designed semantic resources such as WordNet or FrameNet. By adding WordNet hypernyms, POS and NER tags our model outperforms the state of the art that uses significantly more resources. The code for our model is available at `www.socher.org`.

## 2 MV-RNN: A Recursive Matrix-Vector Model

The dominant approach for building representations of multi-word units from single word vector representations has been to form a linear combination of the single word representations, such as a sum or weighted average. This happens in information retrieval and in various text similarity functions based on lexical similarity. These approaches can work well when the meaning of a text is literally "the sum of its parts", but fails when words function as operators that modify the meaning of another word: the meaning of "extremely strong" cannot be captured as the sum of word representations for "extremely" and "strong."

The model of Socher et al. (2011c) provided a new possibility for moving beyond a linear combination, through use of a matrix $W$ that multiplied the word vectors $(a, b)$, and a nonlinearity function $g$ (such as a sigmoid or $\tanh$). They compute the parent vector $p$ that describes both words as

$$p = g\left(W \begin{bmatrix} a \\ b \end{bmatrix}\right) \quad (1)$$

and apply this function recursively inside a binarized parse tree so that it can compute vectors for multi-word sequences. Even though the nonlinearity allows to express a wider range of functions, it is almost certainly too much to expect a single fixed $W$ matrix to be able to capture the meaning combination effects of all natural language operators. After all, inside the function $g$, we have the same linear transformation for all possible pairs of word vectors.

Recent work has started to capture the behavior of natural language operators inside semantic vector spaces by modeling them as matrices, which would allow a matrix for "extremely" to appropriately modify vectors for "smelly" or "strong" (Baroni and Zamparelli, 2010; Zanzotto et al., 2010). These approaches are along the right lines but so far have been restricted to capture linear functions of pairs of words whereas we would like nonlinear functions to compute compositional meaning representations for multi-word phrases or full sentences.

The MV-RNN combines the strengths of both of these ideas by (i) assigning a vector and a matrix to *every* word and (ii) learning an input-specific, nonlinear, compositional function for computing vector and matrix representations for multi-word sequences of any syntactic type. Assigning vector-matrix representations to all words instead of only to words of one part of speech category allows for greater flexibility which benefits performance. If a word lacks operator semantics, its matrix can be an identity matrix. However, if a word acts mainly as an operator,

such as "extremely", its vector can become close to zero, while its matrix gains a clear operator meaning, here magnifying the meaning of the modified word in both positive and negative directions.

In this section we describe the initial word representations, the details of combining two words as well as the multi-word extensions. This is followed by an explanation of our training procedure.

### 2.1 Matrix-Vector Neural Word Representation

We represent a word as both a continuous vector and a matrix of parameters. We initialize all word vectors $x \in \mathbb{R}^n$ with pre-trained 50-dimensional word vectors from the unsupervised model of Collobert and Weston (2008). Using Wikipedia text, their model learns word vectors by predicting how likely it is for each word to occur in its context. Similar to other local co-occurrence based vector space models, the resulting word vectors capture syntactic and semantic information. Every word is also associated with a matrix $X$. In all experiments, we initialize matrices as $X = I + \epsilon$, i.e., the identity plus a small amount of Gaussian noise. If the vectors have dimensionality $n$, then each word's matrix has dimensionality $X \in \mathbb{R}^{n \times n}$. While the initialization is random, the vectors and matrices will subsequently be modified to enable a sequence of words to compose a vector that can predict a distribution over semantic labels. Henceforth, we represent any phrase or sentence of length $m$ as an ordered list of vector-matrix pairs $((a, A), \ldots, (m, M))$, where each pair is retrieved based on the word at that position.

### 2.2 Composition Models for Two Words

We first review composition functions for two words. In order to compute a parent vector $p$ from two consecutive words and their respective vectors $a$ and $b$, Mitchell and Lapata (2010) give as their most general function: $p = f(a, b, R, K)$,where $R$ is the a-priori known syntactic relation and $K$ is background knowledge.

There are many possible functions $f$. For our models, there is a constraint on $p$ which is that it has the same dimensionality as each of the input vectors. This way, we can compare $p$ easily with its children and $p$ can be the input to a composition with another word. The latter is a requirement that will become clear in the next section. This excludes tensor products which were outperformed by simpler weighted addition and multiplication methods in (Mitchell and Lapata, 2010).

We will explore methods that do not require any manually designed semantic resources as background knowledge $K$. No explicit knowledge about the type of relation $R$ is used. Instead we want the model to capture this implicitly via the learned matrices. We propose the following combination function which is input dependent:

$$p = f_{A,B}(a, b) = f(Ba, Ab) = g\left(W \begin{bmatrix} Ba \\ Ab \end{bmatrix}\right), \quad (2)$$

where $A, B$ are matrices for single words, the global $W \in \mathbb{R}^{n \times 2n}$ is a matrix that maps both transformed words back into the same $n$-dimensional space. The element-wise function $g$ could be simply the identity function but we use instead a nonlinearity such as the sigmoid or hyperbolic tangent $\tanh$. Such a nonlinearity will allow us to approximate a wider range of functions beyond purely linear functions. We can also add a bias term before applying $g$ but omit this for clarity. Rewriting the two transformed vectors as one vector $z$, we get $p = g(Wz)$ which is a single layer neural network. In this model, the word matrices can capture compositional effects specific to each word, whereas $W$ captures a general composition function.

This function builds upon and generalizes several recent models in the literature. The most related work is that of (Mitchell and Lapata, 2010; Zanzotto et al., 2010) who introduced and explored the composition function $p = Ba + Ab$ for word pairs. This model is a special case of Eq. 2 when we set $W = [II]$ (i.e. two concatenated identity matrices) and $g(x) = x$ (the identity function). Baroni and Zamparelli (2010) computed the parent vector of adjective-noun pairs by $p = Ab$, where $A$ is an adjective matrix and $b$ is a vector for a noun. This cannot capture nouns modifying other nouns, e.g., *disk drive*. This model too is a special case of the above model with $B = 0_{n \times n}$. Lastly, the models of (Socher et al., 2011b; Socher et al., 2011c; Socher et al., 2011a) as described above are also special cases with both $A$ and $B$ set to the identity matrix. We will compare to these special cases in our experiments.

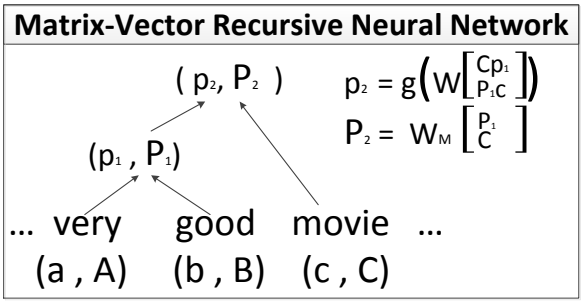


Figure 2: Example of how the MV-RNN merges a phrase with another word at a nonterminal node of a parse tree.

### 2.3 Recursive Compositions of Multiple Words and Phrases

This section describes how we extend a word-pair matrix-vector-based compositional model to learn vectors and matrices for longer sequences of words. The main idea is to apply the same function $f$ to pairs of constituents in a parse tree. For this to work, we need to take as input a binary parse tree of a phrase or sentence and also compute matrices at each nonterminal parent node. The function $f$ can be readily used for phrase vectors since it is recursively compatible ($p$ has the same dimensionality as its children). For computing nonterminal phrase matrices, we define the function

$$P = f_M(A, B) = W_M \begin{bmatrix} A \\ B \end{bmatrix}, \qquad (3)$$

where $W_M \in \mathbb{R}^{n \times 2n}$, so $P \in \mathbb{R}^{n \times n}$ just like each input matrix.

After two words form a constituent in the parse tree, this constituent can now be merged with another one by applying the same functions $f$ and $f_M$. For instance, to compute the vectors and matrices depicted in Fig. 2, we first merge words $a$ and $b$ and their matrices: $p_1 = f(Ba, Ab), P_1 = f_M(A, B)$. The resulting vector-matrix pair $(p_1, P_1)$ can now be used to compute the full phrase when combining it with word $c$ and computing $p_2 = f(Cp_1, P_1 c), P_2 = f_M(P_1, C)$. The model computes vectors and matrices in a bottom-up fashion, applying the functions $f, f_M$ to its own previous output (i.e. recursively) until it reaches the top node of the tree which represents the entire sentence.

For experiments with longer sequences we will compare to standard RNNs and the special case of the MV-RNN that computes the parent by $p = Ab + Ba$, which we name the *linear Matrix-Vector Recursion* model (linear MVR). Previously, this model had not been trained for multi-word sequences. Sec. 6 talks about alternatives for compositionality.

### 2.4 Objective Functions for Training

One of the advantages of RNN-based models is that each node of a tree has associated with it a distributed vector representation (the parent vector $p$) which can also be seen as features describing that phrase. We train these representations by adding on top of each parent node a simple $\mathrm{softmax}$ classifier to predict a class distribution over, e.g., sentiment or relationship classes: $d(p) = \mathrm{softmax}(W^{label} p)$. If there are $K$ labels, then $d \in \mathbb{R}^K$ is a $K$-dimensional multinomial distribution. For the applications below (excluding logic), the corresponding error function $E(s, t, \theta)$ that we minimize for a sentence $s$ and its tree $t$ is the sum of cross-entropy errors at all nodes.

The only other methods that use this type of objective function are (Socher et al., 2011b; Socher et al., 2011c), who also combine it with either a score or reconstruction error. Hence, for comparisons to other related work, we need to merge variations of computing the parent vector $p$ with this classifier. The main difference is that the MV-RNN has more flexibility since it has an input specific recursive function $f_{A,B}$ to compute each parent. In the following applications, we will use the $\mathrm{softmax}$ classifier to predict both sentiment distributions and noun-noun relationships.

### 2.5 Learning

Let $\theta = (W, W_M, W^{label}, L, L_M)$ be our model parameters and $\lambda$ a vector with regularization hyperparameters for all model parameters. $L$ and $L_M$ are the sets of all word vectors and word matrices. The gradient of the overall objective function $J$ becomes:

$$\frac{\partial J}{\partial \theta} = \frac{1}{N} \sum_{(x,t)} \frac{\partial E(x, t; \theta)}{\partial \theta} + \lambda \theta. \qquad (4)$$

To compute this gradient, we first compute all tree nodes $(p_i, P_i)$ from the bottom-up and then take derivatives of the $\mathrm{softmax}$ classifiers at each node in the tree from the top down. Derivatives are computed efficiently via backpropagation through structure (Goller and Küchler, 1996). Even though the

objective is not convex, we found that L-BFGS run over the complete training data (batch mode) minimizes the objective well in practice and convergence is smooth. For more information see (Socher et al., 2010).

### 2.6 Low-Rank Matrix Approximations

If every word is represented by an $n$-dimensional vector and additionally by an $n \times n$ matrix, the dimensionality of the whole model may become too large with commonly used vector sizes of $n = 100$. In order to reduce the number of parameters, we represent word matrices by the following low-rank plus diagonal approximation:

$$A = UV + \mathrm{diag}(a), \tag{5}$$

where $U \in \mathbb{R}^{n \times r}, V \in \mathbb{R}^{r \times n}, a \in \mathbb{R}^n$ and we set the rank for all experiments to $r = 3$.

### 2.7 Discussion: Evaluation and Generality

Evaluation of compositional vector spaces is a complex task. Most related work compares similarity judgments of unsupervised models to those of human judgments and aims at high correlation. These evaluations can give important insights. However, even with good correlation the question remains how these models would perform on downstream NLP tasks such as sentiment detection. We experimented with unsupervised learning of general vector-matrix representations by having the MV-RNN predict words in their correct context. Initializing the models with these general representations, did not improve the performance on the tasks we consider. For sentiment analysis, this is not surprising since antonyms often get similar vectors during unsupervised learning from co-occurrences due to high similarity of local syntactic contexts. In our experiments, the high prediction performance came from supervised learning of meaning representations using labeled data. While these representations are task-specific, they could be used across tasks in a multi-task learning setup. However, in order to fairly compare to related work, we use only the supervised data of each task. Before we describe our full-scale experiments, we analyze the model's expressive powers.

## 3 Model Analysis

This section analyzes the model with two proof-of-concept studies. First, we examine its ability to learn operator semantics for adverb-adjective pairs. If a model cannot correctly capture how an adverb operates on the meaning of adjectives, then there's little chance it can learn operators for more complex relationships. The second study analyzes whether the MV-RNN can learn simple boolean operators of propositional logic such as conjunctives or negation from truth values. Again, if a model did not have this ability, then there's little chance it could learn these frequently occurring phenomena from the noisy language of real texts such as movie reviews.

### 3.1 Predicting Sentiment Distributions of Adverb-Adjective Pairs

The first study considers the prediction of fine-grained sentiment distributions of adverb-adjective pairs and analyzes different possibilities for computing the parent vector $p$. The results show that the MV-RNN operators are powerful enough to capture the operational meanings of various types of adverbs. For example, *very* is an intensifier, *pretty* is an attenuator, and *not* can negate or strongly attenuate the positivity of an adjective. For instance *not great* is still *pretty good* and not *terrible*; see Potts (2010) for details.

We use a publicly available IMDB dataset of extracted adverb-adjective pairs from movie reviews.[1] The dataset provides the distribution over star ratings: Each consecutive word pair appears a certain number of times in reviews that have also associated with them an overall rating of the movie. After normalizing by the total number of occurrences, one gets a multinomial distribution over ratings. Only word pairs that appear at least 50 times are kept. Of the remaining pairs, we use 4211 randomly sampled ones for training and a separate set of 1804 for testing. We never give the algorithm sentiment distributions for single words, and, while single words overlap between training and testing, the test set consists of never before seen word pairs.

The $\mathrm{softmax}$ classifier is trained to minimize the cross entropy error. Hence, an evaluation in terms of KL-divergence is the most reasonable choice. It is

[1] *http://compprag.christopherpotts.net/reviews.html*

| Method | Avg KL |
|---|---|
| Uniform | 0.327 |
| Mean train | 0.193 |
| $p = \frac{1}{2}(a + b)$ | 0.103 |
| $p = a \otimes b$ | 0.103 |
| $p = [a; b]$ | 0.101 |
| $p = Ab$ | 0.103 |
| RNN | 0.093 |
| Linear MVR | 0.092 |
| MV-RNN | **0.091** |

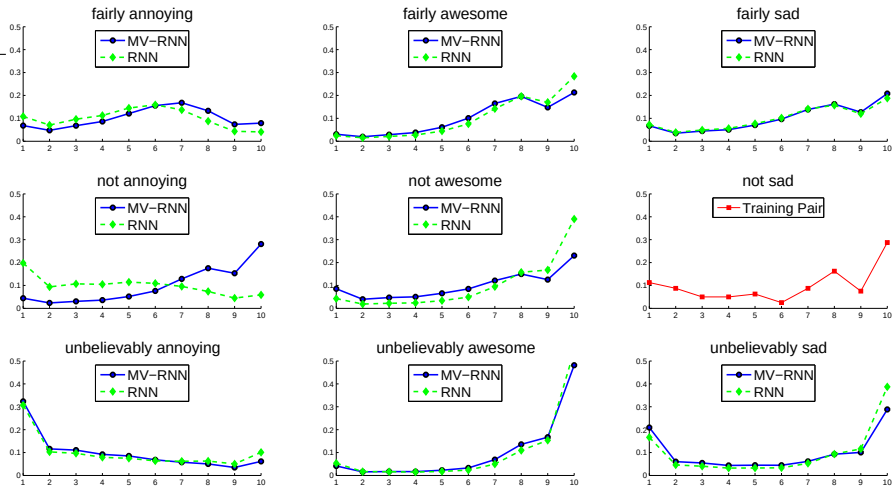


Figure 3: **Left**: Average KL-divergence for predicting sentiment distributions of unseen adverb-adjective pairs of the test set. See text for $p$ descriptions. Lower is better. The main difference in the KL divergence comes from the few negation pairs in the test set. **Right**: Predicting sentiment distributions (over 1-10 stars on the $x$-axis) of adverb-adjective pairs. Each row has the same adverb and each column the same adjective. Many predictions are similar between the two models. The RNN and linear MVR are not able to modify the sentiment correctly: *not awesome* is more positive than *fairly awesome* and *not annoying* has a similar shape as *unbelievably annoying*. Predictions of the linear MVR model are almost identical to the standard RNN for these examples.

defined as $KL(g||p) = \sum_i g_i \log(g_i/p_i)$, where $g$ is the gold distribution and $p$ is the predicted one.

We compare to several baselines and ablations of the MV-RNN model. An (adverb,adjective) pair is described by its vectors $(a, b)$ and matrices $(A, B)$.

1 $p = 0.5(a + b)$, vector average
2. $p = a \otimes b$, element-wise vector multiplication
3. $p = [a; b]$, vector concatenation
4. $p = Ab$, similar to (Baroni and Lenci, 2010)
5. $p = g(W[a; b])$, RNN, similar to Socher et al.
6. $p = Ab + Ba$, Linear MVR, similar to (Mitchell and Lapata, 2010; Zanzotto et al., 2010)
7. $p = g(W[Ba; Ab])$, MV-RNN

The final distribution is always predicted by a $\mathrm{softmax}$ classifier whose inputs $p$ vary for each of the models. This objective function (see Sec. 2.4) is different to all previously published work except that of (Socher et al., 2011c).

We cross-validated all models over regularization parameters for word vectors, the $\mathrm{softmax}$ classifier, the RNN parameter $W$ and the word operators $(10^{-4}, 10^{-3})$ and word vector sizes $(n = 6, 8, 10, 12, 15, 20)$. All models performed best at vector sizes of below 12. Hence, it is the model's power and not the number of parameters that determines the performance. The table in Fig. 3 shows the average KL-divergence on the test set. It shows that the idea of matrix-vector representations for all words and having a nonlinearity are both important. The MV-RNN which combines these two ideas is best able to learn the various compositional effects. The main difference in KL divergence comes from the few negation cases in the test set. Fig. 3 shows examples of predicted distributions. Many of the predictions are accurate and similar between the top models. However, only the MV-RNN has enough expressive power to allow negation to completely shift the sentiment with respect to an adjective. A negated adjective carrying negative sentiment becomes slightly positive, whereas *not awesome* is correctly attenuated. All three top models correctly capture the U-shape of *unbelievably sad*. This pair peaks at both the negative and positive spectrum because it is ambiguous. When referring to the performance of actors, it is very negative, but, when talking about the plot, many people enjoy sad and thought-provoking movies. The $p = Ab$ model does not perform well because it cannot model the fact that for an adjective like "sad," the operator of "unbelievably" behaves differently.

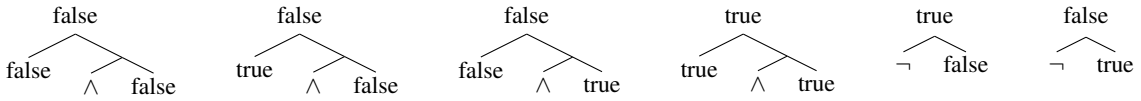


Figure 4: Training trees for the MV-RNN to learn propositional operators. The model learns vectors and operators for $\wedge$ (*and*) and $\neg$ (negation). The model outputs the exact representations of *false* and *true* respectively at the top node. Hence, the operators can be combined recursively an arbitrary number of times for more complex logical functions.

### 3.2 Logic- and Vector-based Compositionality

Another natural question is whether the MV-RNN can, in general, capture some of the simple boolean logic that is sometimes found in language. In other words, can it learn some of the propositional logic operators such as *and, or, not* in terms of vectors and matrices from a few examples. Answering this question can also be seen as a first step towards bridging the gap between logic-based, formal semantics (Montague, 1974) and vector space models.

The logic-based view of language accounts nicely for compositionality by directly mapping syntactic constituents to lambda calculus expressions. At the word level, the focus is on function words, and nouns and adjectives are often defined only in terms of the sets of entities they denote in the world. Most words are treated as atomic symbols with no relation to each other. There have been many attempts at automatically parsing natural language to a logical form using recursive compositional rules.

Conversely, vector space models have the attractive property that they can automatically extract knowledge from large corpora without supervision. Unlike logic-based approaches, these models allow us to make fine-grained statements about the semantic similarity of words which correlate well with human judgments (Griffiths et al., 2007). Logic-based approaches are often seen as orthogonal to distributional vector-based approaches. However, Garrette et al. (2011) recently introduced a combination of a vector space model inside a Markov Logic Network.

One open question is whether vector-based models can learn some of the simple logic encountered in language such as negation or conjunctives. To this end, we illustrate in a simple example that our MV-RNN model and its learned word matrices (operators) have the ability to learn propositional logic operators such as $\wedge, \vee, \neg$ (and, or, not). This is a necessary (though not sufficient) condition for the ability to pick up these phenomena in real datasets and tasks such as sentiment detection which we focus on in the subsequent sections.

Our setup is as follows. We train on 6 strictly right-branching trees as in Fig. 4. We consider the 1-dimensional case and fix the representation for *true* to $(t = 1, T = 1)$ and *false* to $(f = 0, F = 1)$. Fixing the operators to the $1 \times 1$ identity matrix 1 is essentially ignoring them. The objective is then to create a perfect reconstruction of $(t, T)$ or $(f, F)$ (depending on the formula), which we achieve by the least squares error between the top vector's representation and the corresponding truth value, e.g. for $\neg false$: $\min ||p_{top} - t||^2 + ||P_{top} - T||^2$.

As our function $g$ (see Eq. 2), we use a linear threshold unit: $g(x) = \max(\min(x, 1), 0)$. Giving the derivatives computed for the objective function for the examples in Fig. 4 to a standard L-BFGS optimizer quickly yields a training error of 0. Hence, the output of these 6 examples has exactly one of the truth representations, making it recursively compatible with further combinations of operators. Thus, we can combine these operators to construct any propositional logic function of any number of inputs (including xor). Hence, this MV-RNN is complete in terms of propositional logic.

## 4 Predicting Movie Review Ratings

In this section, we analyze the model's performance on full length sentences. We compare to previous state of the art methods on a standard benchmark dataset of movie reviews (Pang and Lee, 2005; Nakagawa et al., 2010; Socher et al., 2011c). This dataset consists of 10,000 positive and negative single sentences describing movie sentiment. In this and the next experiment we use binarized trees from the Stanford Parser (Klein and Manning, 2003). We use the exact same setup and parameters (regularization, word vector size, etc.) as the published code of Socher et al. (2011c).[2]

[2] www.socher.org

| Method | Acc. |
|---|---|
| Tree-CRF (Nakagawa et al., 2010) | 77.3 |
| RAE (Socher et al., 2011c) | 77.7 |
| Linear MVR | 77.1 |
| MV-RNN | **79.0** |

Table 1: Accuracy of classification on full length movie review polarity (MR).

| S. | C. | Review sentence |
|---|---|---|
| 1 | √ | The film is bright and flashy in all the right ways. |
| 0 | √ | Not always too whimsical for its own good this strange hybrid of crime thriller, quirky character study, third-rate romance and female empowerment fantasy never really finds the tonal or thematic glue it needs. |
| 0 | √ | Doesn't come close to justifying the hype that surrounded its debut at the Sundance film festival two years ago. |
| 0 | x | Director Hoffman, his writer and Kline's agent should serve detention. |
| 1 | x | A bodice-ripper for intellectuals. |

Table 2: Hard movie review examples of positive (1) and negative (0) sentiment (S.) that of all methods only the MV-RNN predicted correctly (C: √) or could not classify as correct either (C: x).

Table 1 shows comparisons to the system of (Nakagawa et al., 2010), a dependency tree based classification method that uses CRFs with hidden variables. The state of the art recursive autoencoder model of Socher et al. (2011c) obtained 77.7% accuracy. Our new MV-RNN gives the highest performance, outperforming also the linear MVR (Sec. 2.2).

Table 2 shows several hard examples that only the MV-RNN was able to classify correctly. None of the methods correctly classified the last two examples which require more world knowledge.

## 5 Classification of Semantic Relationships

The previous task considered global classification of an entire phrase or sentence. In our last experiment we show that the MV-RNN can also learn how a syntactic context composes an aggregate meaning of the semantic relationships between words. In particular, the task is finding semantic relationships between pairs of nominals. For instance, in the sentence "My [apartment]$_{e1}$ has a pretty large [kitchen]$_{e2}$.", we want to predict that the kitchen and apartment are in a *component-whole* relationship. Predicting such

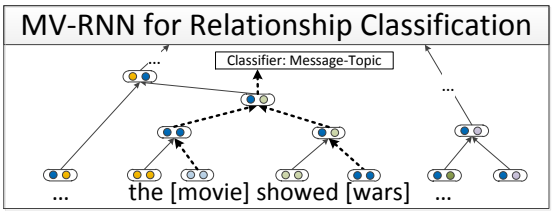


Figure 5: The MV-RNN learns vectors in the path connecting two words (dotted lines) to determine their semantic relationship. It takes into consideration a variable length sequence of various word types in that path.

semantic relations is useful for information extraction and thesaurus construction applications. Many approaches use features for all words on the path between the two words of interest. We show that by building a single compositional semantics for the minimal constituent including both terms one can achieve a higher performance.

This task requires the ability to deal with sequences of words of arbitrary type and length in between the two nouns in question.Fig. 5 explains our method for classifying nominal relationships. We first find the path in the parse tree between the two words whose relation we want to classify. We then select the highest node of the path and classify the relationship using that node's vector as features. We apply the same type of MV-RNN model as in sentiment to the subtree spanned by the two words.

We use the dataset and evaluation framework of SemEval-2010 Task 8 (Hendrickx et al., 2010). There are 9 ordered relationships (with two directions) and an undirected *other* class, resulting in 19 classes. Among the relationships are: message-topic, cause-effect, instrument-agency (etc. see Table 3 for list). A pair is counted as correct if the order of the words in the relationship is correct.

Table 4 lists results for several competing methods together with the resources and features used by each method. We compare to the systems of the competition which are described in Hendrickx et al. (2010) as well as the RNN and linear MVR. Most systems used a considerable amount of hand-designed semantic resources. In contrast to these methods, the MV-RNN only needs a parser for the tree structure and learns all semantics from unlabeled corpora and the training data. Only the SemEval training dataset is specific to this task, the re-

| Relationship | Sentence with labeled nouns for which to predict relationships |
|---|---|
| Cause-Effect(e2,e1) | Avian [influenza]$_{e1}$ is an infectious disease caused by type a strains of the influenza [virus]$_{e2}$. |
| Entity-Origin(e1,e2) | The [mother]$_{e1}$ left her native [land]$_{e2}$ about the same time and they were married in that city. |
| Message-Topic(e2,e1) | Roadside [attractions]$_{e1}$ are frequently advertised with [billboards]$_{e2}$ to attract tourists. |
| Product-Producer(e1,e2) | A child is told a [lie]$_{e1}$ for several years by their [parents]$_{e2}$ before he/she realizes that ... |
| Entity-Destination(e1,e2) | The accident has spread [oil]$_{e1}$ into the [ocean]$_{e2}$. |
| Member-Collection(e2,e1) | The siege started, with a [regiment]$_{e1}$ of lightly armored [swordsmen]$_{e2}$ ramming down the gate. |
| Instrument-Agency(e2,e1) | The core of the [analyzer]$_{e1}$ identifies the paths using the constraint propagation [method]$_{e2}$. |
| Component-Whole(e2,e1) | The size of a [tree]$_{e1}$ [crown]$_{e2}$ is strongly correlated with the growth of the tree. |
| Content-Container(e1,e2) | The hidden [camera]$_{e1}$, found by a security guard, was hidden in a business card-sized [leaflet box]$_{e2}$ placed at an unmanned ATM in Tokyo's Minato ward in early September. |

Table 3: Examples of correct classifications of ordered, semantic relations between nouns by the MV-RNN. Note that the final classifier is a recursive, compositional function of all the words in the syntactic path between the bracketed words. The paths vary in length and the words vary in type.

| Classifier | Feature Sets | F1 |
|---|---|---|
| SVM | POS, stemming, syntactic patterns | 60.1 |
| SVM | word pair, words in between | 72.5 |
| SVM | POS, WordNet, stemming, syntactic patterns | 74.8 |
| SVM | POS, WordNet, morphological features, thesauri, Google $n$-grams | 77.6 |
| MaxEnt | POS, WordNet, morphological features, noun compound system, thesauri, Google $n$-grams | 77.6 |
| SVM | POS, WordNet, prefixes and other morphological features, POS, dependency parse features, Levin classes, PropBank, FrameNet, NomLex-Plus, Google $n$-grams, paraphrases, TextRunner | 82.2 |
| RNN | - | 74.8 |
| Lin.MVR | - | 73.0 |
| MV-RNN | - | 79.1 |
| RNN | POS,WordNet,NER | 77.6 |
| Lin.MVR | POS,WordNet,NER | 78.7 |
| MV-RNN | POS,WordNet,NER | **82.4** |

Table 4: Learning methods, their feature sets and F1 results for predicting semantic relations between nouns. The MV-RNN outperforms all but one method without any additional feature sets. By adding three such features, it obtains state of the art performance.

maining inputs and the training setup are the same as in previous sentiment experiments.

The best method on this dataset (Rink and Harabagiu, 2010) obtains 82.2% F1. In order to see whether our system can improve over this system, we added three features to the MV-RNN vector and trained another $\mathrm{softmax}$ classifier. The features and their performance increases were POS tags (+0.9); WordNet hypernyms (+1.3) and named entity tags (NER) of the two words (+0.6). Features were computed using the code of Ciaramita and Altun (2006).[3] With these features, the performance improved over the state of the art system. Table 3 shows random correct classification examples.

## 6 Related work

Distributional approaches have become omnipresent for the recognition of semantic similarity between words and the treatment of compositionality has seen much progress in recent years. Hence, we cannot do justice to the large amount of literature. Commonly, single words are represented as vectors of distributional characteristics – e.g., their frequencies in specific syntactic relations or their co-occurrences with given context words (Pado and Lapata, 2007; Baroni and Lenci, 2010; Turney and Pantel, 2010). These representations have proven very effective in sense discrimination and disambiguation (Schütze, 1998), automatic thesaurus extraction (Lin, 1998; Curran, 2004) and selectional preferences.

There are several sophisticated ideas for compositionality in vector spaces. Mitchell and Lapata (2010) present an overview of the most important compositional models, from simple vector addition and component-wise multiplication to tensor products, and convolution (Metcalfe, 1990). They measured the similarity between word pairs such as compound nouns or verb-object pairs and compared these with human similarity judgments. Simple vector averaging or multiplication performed best, hence our focus on related baselines above.

[3] sourceforge.net/projects/supersensetag/

Other important models are tensor products (Clark and Pulman, 2007), quantum logic (Widdows, 2008), holographic reduced representations (Plate, 1995) and the Compositional Matrix Space model (Rudolph and Giesbrecht, 2010). RNNs are related to autoencoder models such as the recursive autoassociative memory (RAAM) (Pollack, 1990) or recurrent neural networks (Elman, 1991). Bottou (2011) and Hinton (1990) discussed related models such as recursive autoencoders for text understanding.

Our model builds upon and generalizes the models of (Mitchell and Lapata, 2010; Baroni and Zamparelli, 2010; Zanzotto et al., 2010; Socher et al., 2011c) (see Sec. 2.2). We compare to them in our experiments. Yessenalina and Cardie (2011) introduce a sentiment analysis model that describes words as matrices and composition as matrix multiplication. Since matrix multiplication is associative, this cannot capture different scopes of negation or syntactic differences. Their model, is a special case of our encoding model (when you ignore vectors, fix the tree to be strictly branching in one direction and use as the matrix composition function $P = AB$). Since our classifiers are trained on the vectors, we cannot compare to this approach directly. Grefenstette and Sadrzadeh (2011) learn matrices for verbs in a categorical model. The trained matrices improve correlation with human judgments on the task of identifying relatedness of subject-verb-object triplets.

## 7 Conclusion

We introduced a new model towards a complete treatment of compositionality in word vector spaces. Our model builds on a syntactically plausible parse tree and can handle compositional phenomena. The main novelty of our model is the combination of matrix-vector representations with a recursive neural network. It can learn both the meaning vectors of a word and how that word modifies its neighbors (via its matrix). The MV-RNN combines attractive theoretical properties with good performance on large, noisy datasets. It generalizes several models in the literature, can learn propositional logic, accurately predicts sentiment and can be used to classify semantic relationships between nouns in a sentence.

## Acknowledgments

We thank for great discussions about the paper: John Platt, Chris Potts, Josh Tenenbaum, Mihai Surdeanu, Quoc Le and Kevin Miller. The authors gratefully acknowledges the support of the Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181, and the DARPA Deep Learning program under contract number FA8650-10-C-7020. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of DARPA, AFRL, or the US government.

## References

M. Baroni and A. Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721.

M. Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *EMNLP*.

L. Bottou. 2011. From machine learning to machine reasoning. *CoRR*, abs/1102.1808.

M. Ciaramita and Y. Altun. 2006. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *EMNLP*.

S. Clark and S. Pulman. 2007. Combining symbolic and distributional models of meaning. In *Proceedings of the AAAI Spring Symposium on Quantum Interaction*, pages 52–55.

R. Collobert and J. Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *ICML*.

J. Curran. 2004. *From Distributional to Semantic Similarity*. Ph.D. thesis, University of Edinburgh.

J. L. Elman. 1991. Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning*, 7(2-3).

G. Frege. 1892. Über Sinn und Bedeutung. In *Zeitschrift für Philosophie und philosophische Kritik, 100*.

D. Garrette, K. Erk, and R. Mooney. 2011. Integrating Logical Representations with Probabilistic Information using Markov Logic. In *Proceedings of the International Conference on Computational Semantics*.

C. Goller and A. Küchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *Proceedings of the International Conference on Neural Networks (ICNN-96)*.

E. Grefenstette and M. Sadrzadeh. 2011. Experimental support for a categorical compositional distributional model of meaning. In *EMNLP*.

T. L. Griffiths, J. B. Tenenbaum, and M. Steyvers. 2007. Topics in semantic representation. *Psychological Review*, 114.

I. Hendrickx, S.N. Kim, Z. Kozareva, P. Nakov, D. Ó Séaghdha, S. Padó, M. Pennacchiotti, L. Romano, and S. Szpakowicz. 2010. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation*.

G. E. Hinton. 1990. Mapping part-whole hierarchies into connectionist networks. *Artificial Intelligence*, 46(1-2).

R. Jones, B. Rey, O. Madani, and W. Greiner. 2006. Generating query substitutions. In *Proceedings of the 15th international conference on World Wide Web*.

D. Klein and C. D. Manning. 2003. Accurate unlexicalized parsing. In *ACL*.

D. Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of COLING-ACL*, pages 768–774.

E. J. Metcalfe. 1990. A compositive holographic associative recall model. *Psychological Review*, 88:627–661.

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

R. Montague. 1974. English as a formal language. *Linguaggi nella Societa e nella Tecnica*, pages 189–224.

T. Nakagawa, K. Inui, and S. Kurohashi. 2010. Dependency tree-based sentiment classification using CRFs with hidden variables. In *NAACL*, HLT.

M. Paşca, D. Lin, J. Bigham, A. Lifchits, and A. Jain. 2006. Names and similarities on the web: fact extraction in the fast lane. In *ACL*.

S. Pado and M. Lapata. 2007. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199.

B. Pang and L. Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*, pages 115–124.

T. A. Plate. 1995. Holographic reduced representations. *IEEE Transactions on Neural Networks*, 6(3):623–641.

J. B. Pollack. 1990. Recursive distributed representations. *Artificial Intelligence*, 46, November.

C. Potts. 2010. On the negativity of negation. In David Lutz and Nan Li, editors, *Proceedings of Semantics and Linguistic Theory 20*. CLC Publications, Ithaca, NY.

L. Ratinov, D. Roth, D. Downey, and M. Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *ACL*.

B. Rink and S. Harabagiu. 2010. UTD: Classifying semantic relations by combining lexical and semantic resources. In *Proceedings of the 5th International Workshop on Semantic Evaluation*.

S. Rudolph and E. Giesbrecht. 2010. Compositional matrix-space models of language. In *ACL*.

H. Schütze. 1998. Automatic word sense discrimination. *Computational Linguistics*, 24:97–124.

R. Socher, C. D. Manning, and A. Y. Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*.

R. Socher, E. H. Huang, J. Pennington, A. Y. Ng, and C. D. Manning. 2011a. Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection. In *NIPS*. MIT Press.

R. Socher, C. Lin, A. Y. Ng, and C.D. Manning. 2011b. Parsing Natural Scenes and Natural Language with Recursive Neural Networks. In *ICML*.

R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning. 2011c. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. In *EMNLP*.

P. D. Turney and P. Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

D. Widdows. 2008. Semantic vector products: Some initial investigations. In *Proceedings of the Second AAAI Symposium on Quantum Interaction*.

A. Yessenalina and C. Cardie. 2011. Compositional matrix-space models for sentiment analysis. In *EMNLP*.

F.M. Zanzotto, I. Korkontzelos, F. Fallucchi, and S. Manandhar. 2010. Estimating linear models for compositional distributional semantics. COLING.