Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank

Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang,
Christopher D. Manning, Andrew Y. Ng and Christopher Potts
Stanford University, Stanford, CA 94305, USA
richard@socher.org,{aperelyg,jcchuang,ang}@cs.stanford.edu
{jeaneis,manning,cgpotts}@stanford.edu

Abstract

Semantic word spaces have been very useful but cannot express the meaning of longer phrases in a principled way. Further progress towards understanding compositionality in tasks such as sentiment detection requires richer supervised training and evaluation resources and more powerful models of composition. To remedy this, we introduce a Sentiment Treebank. It includes fine grained sentiment labels for 215,154 phrases in the parse trees of 11,855 sentences and presents new challenges for sentiment compositionality. To address them, we introduce the Recursive Neural Tensor Network. When trained on the new treebank, this model outperforms all previous methods on several metrics. It pushes the state of the art in single sentence positive/negative classification from 80% up to 85.4%. The accuracy of predicting fine-grained sentiment labels for all phrases reaches 80.7%, an improvement of 9.7% over bag of features baselines. Lastly, it is the only model that can accurately capture the effects of negation and its scope at various tree levels for both positive and negative phrases.

1 Introduction

Semantic vector spaces for single words have been widely used as features (Turney and Pantel, 2010). Because they cannot capture the meaning of longer phrases properly, compositionality in semantic vector spaces has recently received a lot of attention (Mitchell and Lapata, 2010; Socher et al., 2010; Zanzotto et al., 2010; Yessenalina and Cardie, 2011; Socher et al., 2012; Grefenstette et al., 2013). However, progress is held back by the current lack of large and labeled compositionality resources and models to accurately capture the underlying phenomena presented in such data. To address this need, we introduce the Stanford Sentiment Treebank and a powerful Recursive Neural Tensor Network that can accurately predict the compositional semantic effects present in this new corpus.

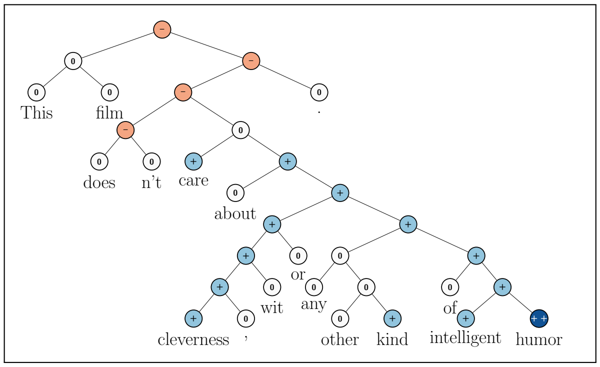


Figure 1: Example of the Recursive Neural Tensor Network accurately predicting 5 sentiment classes, very negative to very positive (– –, –, 0, +, + +), at every node of a parse tree and capturing the negation and its scope in this sentence.

The *Stanford Sentiment Treebank* is the first corpus with fully labeled parse trees that allows for a complete analysis of the compositional effects of sentiment in language. The corpus is based on the dataset introduced by Pang and Lee (2005) and consists of 11,855 single sentences extracted from movie reviews. It was parsed with the Stanford parser (Klein and Manning, 2003) and includes a total of 215,154 unique phrases from those parse trees, each annotated by 3 human judges. This new dataset allows us to analyze the intricacies of sentiment and to capture complex linguistic phenomena. Fig. 1 shows one of the many examples with clear compositional structure. The granularity and size of

Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1631–1642,
Seattle, Washington, USA, 18-21 October 2013. ©2013 Association for Computational Linguistics

this dataset will enable the community to train compositional models that are based on supervised and structured machine learning techniques. While there are several datasets with document and chunk labels available, there is a need to better capture sentiment from short comments, such as Twitter data, which provide less overall signal per document.

In order to capture the compositional effects with higher accuracy, we propose a new model called the Recursive Neural Tensor Network (RNTN). Recursive Neural Tensor Networks take as input phrases of any length. They represent a phrase through word vectors and a parse tree and then compute vectors for higher nodes in the tree using the same tensor-based composition function. We compare to several supervised, compositional models such as standard recursive neural networks (RNN) (Socher et al., 2011b), matrix-vector RNNs (Socher et al., 2012), and baselines such as neural networks that ignore word order, Naive Bayes (NB), bi-gram NB and SVM. All models get a significant boost when trained with the new dataset but the RNTN obtains the highest performance with 80.7% accuracy when predicting fine-grained sentiment for all nodes. Lastly, we use a test set of positive and negative sentences and their respective negations to show that, unlike bag of words models, the RNTN accurately captures the sentiment change and scope of negation. RNTNs also learn that sentiment of phrases following the contrastive conjunction 'but' dominates.

The complete training and testing code, a live demo and the Stanford Sentiment Treebank dataset are available at `http://nlp.stanford.edu/sentiment`.

2 Related Work

This work is connected to five different areas of NLP research, each with their own large amount of related work to which we cannot do full justice given space constraints.

Semantic Vector Spaces. The dominant approach in semantic vector spaces uses distributional similarities of single words. Often, co-occurrence statistics of a word and its context are used to describe each word (Turney and Pantel, 2010; Baroni and Lenci, 2010), such as tf-idf. Variants of this idea use more complex frequencies such as how often a word appears in a certain syntactic context (Pado and Lapata, 2007; Erk and Padó, 2008). However, distributional vectors often do not properly capture the differences in antonyms since those often have similar contexts. One possibility to remedy this is to use neural word vectors (Bengio et al., 2003). These vectors can be trained in an unsupervised fashion to capture distributional similarities (Collobert and Weston, 2008; Huang et al., 2012) but then also be fine-tuned and trained to specific tasks such as sentiment detection (Socher et al., 2011b). The models in this paper can use purely supervised word representations learned entirely on the new corpus.

Compositionality in Vector Spaces. Most of the compositionality algorithms and related datasets capture two word compositions. Mitchell and Lapata (2010) use e.g. two-word phrases and analyze similarities computed by vector addition, multiplication and others. Some related models such as holographic reduced representations (Plate, 1995), quantum logic (Widdows, 2008), discrete-continuous models (Clark and Pulman, 2007) and the recent compositional matrix space model (Rudolph and Giesbrecht, 2010) have not been experimentally validated on larger corpora. Yessenalina and Cardie (2011) compute matrix representations for longer phrases and define composition as matrix multiplication, and also evaluate on sentiment. Grefenstette and Sadrzadeh (2011) analyze subject-verb-object triplets and find a matrix-based categorical model to correlate well with human judgments. We compare to the recent line of work on supervised compositional models. In particular we will describe and experimentally compare our new RNTN model to recursive neural networks (RNN) (Socher et al., 2011b) and matrix-vector RNNs (Socher et al., 2012) both of which have been applied to bag of words sentiment corpora.

Logical Form. A related field that tackles compositionality from a very different angle is that of trying to map sentences to logical form (Zettlemoyer and Collins, 2005). While these models are highly interesting and work well in closed domains and on discrete sets, they could only capture sentiment distributions using separate mechanisms beyond the currently used logical forms.

Deep Learning. Apart from the above mentioned work on RNNs, several compositionality ideas related to neural networks have been discussed by Bottou (2011) and Hinton (1990) and first models such as Recursive Auto-associative memories been experimented with by Pollack (1990). The idea to relate inputs through three way interactions, parameterized by a tensor have been proposed for relation classification (Sutskever et al., 2009; Jenatton et al., 2012), extending Restricted Boltzmann machines (Ranzato and Hinton, 2010) and as a special layer for speech recognition (Yu et al., 2012).

Sentiment Analysis. Apart from the above-mentioned work, most approaches in sentiment analysis use bag of words representations (Pang and Lee, 2008). Snyder and Barzilay (2007) analyzed larger reviews in more detail by analyzing the sentiment of multiple aspects of restaurants, such as food or atmosphere. Several works have explored sentiment compositionality through careful engineering of features or polarity shifting rules on syntactic structures (Polanyi and Zaenen, 2006; Moilanen and Pulman, 2007; Rentoumi et al., 2010; Nakagawa et al., 2010).

3 Stanford Sentiment Treebank

Bag of words classifiers can work well in longer documents by relying on a few words with strong sentiment like 'awesome' or 'exhilarating.' However, sentiment accuracies even for binary positive/negative classification for single sentences has not exceeded 80% for several years. For the more difficult multiclass case including a neutral class, accuracy is often below 60% for short messages on Twitter (Wang et al., 2012). From a linguistic or cognitive standpoint, ignoring word order in the treatment of a semantic task is not plausible, and, as we will show, it cannot accurately classify hard examples of negation. Correctly predicting these hard cases is necessary to further improve performance.

In this section we will introduce and provide some analyses for the new *Sentiment Treebank* which includes labels for every syntactically plausible phrase in thousands of sentences, allowing us to train and evaluate compositional models.

We consider the corpus of movie review excerpts from the `rottentomatoes.com` website originally collected and published by Pang and Lee (2005). The original dataset includes 10,662 sentences, half of which were considered positive and the other half negative. Each label is extracted from a longer movie review and reflects the writer's overall intention for this review. The normalized, lowercased text is first used to recover, from the original website, the text with capitalization. Remaining HTML tags and sentences that are not in English are deleted. The Stanford Parser (Klein and Manning, 2003) is used to parses all 10,662 sentences. In approximately 1,100 cases it splits the snippet into multiple sentences. We then used Amazon Mechanical Turk to label the resulting 215,154 phrases. Fig. 3 shows the interface annotators saw. The slider has 25 different values and is initially set to neutral. The phrases in each hit are randomly sampled from the set of all phrases in order to prevent labels being influenced by what follows. For more details on the dataset collection, see supplementary material.

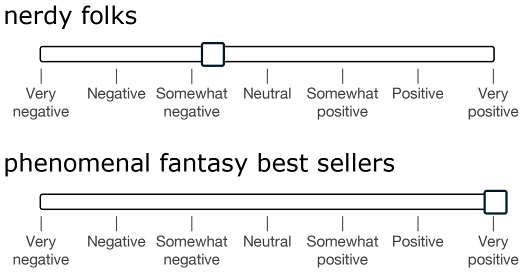


Figure 3: The labeling interface. Random phrases were shown and annotators had a slider for selecting the sentiment and its degree.

Fig. 2 shows the normalized label distributions at each n-gram length. Starting at length 20, the majority are full sentences. One of the findings from labeling sentences based on *reader's perception* is that many of them could be considered neutral. We also notice that stronger sentiment often builds up in longer phrases and the majority of the shorter phrases are neutral. Another observation is that most annotators moved the slider to one of the five positions: negative, somewhat negative, neutral, positive or somewhat positive. The extreme values were rarely used and the slider was not often left in between the ticks. Hence, *even a 5-class classification into these categories captures the main variability of the labels.* We will name this *fine-grained sentiment* classification and our main experiment will be to recover these five labels for phrases of all lengths.

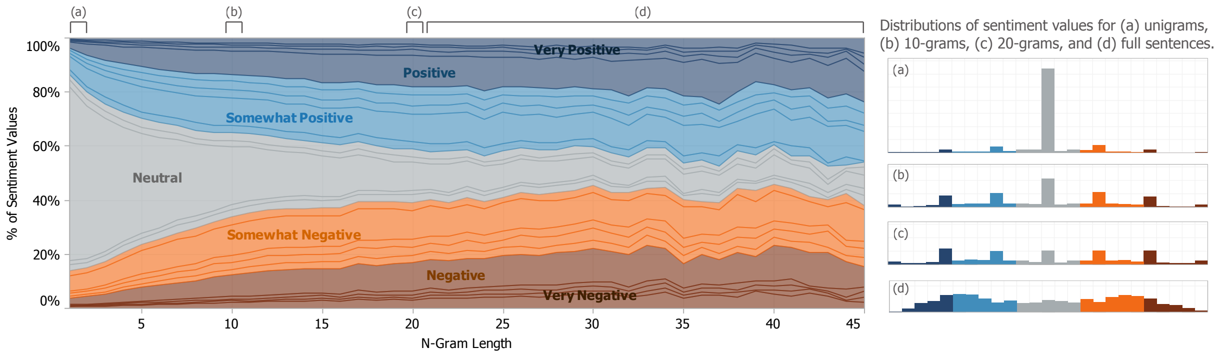


Figure 2: Normalized histogram of sentiment annotations at each n-gram length. Many shorter n-grams are neutral; longer phrases are well distributed. Few annotators used slider positions between ticks or the extreme values. Hence the two strongest labels and intermediate tick positions are merged into 5 classes.

4 Recursive Neural Models

The models in this section compute compositional vector representations for phrases of variable length and syntactic type. These representations will then be used as features to classify each phrase. Fig. 4 displays this approach. When an n-gram is given to the compositional models, it is parsed into a binary tree and each leaf node, corresponding to a word, is represented as a vector. Recursive neural models will then compute parent vectors in a bottom up fashion using different types of compositionality functions g. The parent vectors are again given as features to a classifier. For ease of exposition, we will use the tri-gram in this figure to explain all models.

We first describe the operations that the below recursive neural models have in common: word vector representations and classification. This is followed by descriptions of two previous RNN models and our RNTN.

Each word is represented as a d-dimensional vector. We initialize all word vectors by randomly sampling each value from a uniform distribution: $\mathcal{U}(-r, r)$, where $r = 0.0001$. All the word vectors are stacked in the word embedding matrix $L \in \mathbb{R}^{d \times |V|}$, where $|V|$ is the size of the vocabulary. Initially the word vectors will be random but the L matrix is seen as a parameter that is trained jointly with the compositionality models.

We can use the word vectors immediately as parameters to optimize and as feature inputs to a $\mathrm{softmax}$ classifier. For classification into five classes, we compute the posterior probability over labels given the word vector via:

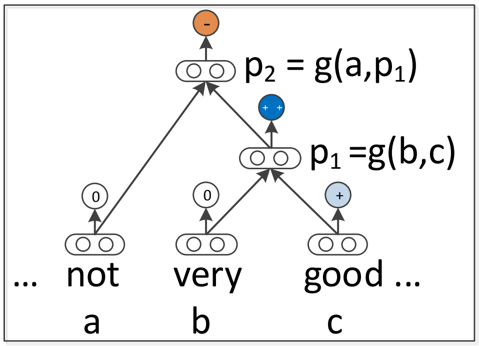


Figure 4: Approach of Recursive Neural Network models for sentiment: Compute parent vectors in a bottom up fashion using a compositionality function g and use node vectors as features for a classifier at that node. This function varies for the different models.

$$y^a = \mathrm{softmax}(W_s a), \quad (1)$$

where $W_s \in \mathbb{R}^{5 \times d}$ is the sentiment classification matrix. For the given tri-gram, this is repeated for vectors b and c. The main task of and difference between the models will be to compute the hidden vectors $p_i \in \mathbb{R}^d$ in a bottom up fashion.

4.1 RNN: Recursive Neural Network

The simplest member of this family of neural network models is the standard recursive neural network (Goller and Küchler, 1996; Socher et al., 2011a). First, it is determined which parent already has all its children computed. In the above tree example, p_1 has its two children's vectors since both are words. RNNs use the following equations to compute the parent vectors:

$$p_1 = f\left(W\begin{bmatrix} b \\ c \end{bmatrix}\right), p_2 = f\left(W\begin{bmatrix} a \\ p_1 \end{bmatrix}\right),$$

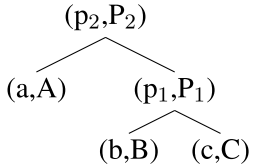
where $f = \tanh$ is a standard element-wise nonlinearity, $W \in \mathbb{R}^{d\times 2d}$ is the main parameter to learn and we omit the bias for simplicity. The bias can be added as an extra column to W if an additional 1 is added to the concatenation of the input vectors. The parent vectors must be of the same dimensionality to be recursively compatible and be used as input to the next composition. Each parent vector p_i, is given to the same $\mathrm{softmax}$ classifier of Eq. 1 to compute its label probabilities.

This model uses the same compositionality function as the recursive autoencoder (Socher et al., 2011b) and recursive auto-associate memories (Pollack, 1990). The only difference to the former model is that we fix the tree structures and ignore the reconstruction loss. In initial experiments, we found that with the additional amount of training data, the reconstruction loss at each node is not necessary to obtain high performance.

4.2 MV-RNN: Matrix-Vector RNN

The MV-RNN is linguistically motivated in that most of the parameters are associated with words and each composition function that computes vectors for longer phrases depends on the actual words being combined. The main idea of the MV-RNN (Socher et al., 2012) is to represent every word and longer phrase in a parse tree as both a vector and a matrix. When two constituents are combined the matrix of one is multiplied with the vector of the other and vice versa. Hence, the compositional function is parameterized by the words that participate in it.

Each word's matrix is initialized as a $d\times d$ identity matrix, plus a small amount of Gaussian noise. Similar to the random word vectors, the parameters of these matrices will be trained to minimize the classification error at each node. For this model, each n-gram is represented as a list of (vector,matrix) pairs, together with the parse tree. For the tree with (vector,matrix) nodes:



the MV-RNN computes the first parent vector and its matrix via two equations:

$$p_1 = f\left(W\begin{bmatrix} Cb \\ Bc \end{bmatrix}\right), P_1 = f\left(W_M\begin{bmatrix} B \\ C \end{bmatrix}\right),$$

where $W_M \in \mathbb{R}^{d\times 2d}$ and the result is again a $d \times d$ matrix. Similarly, the second parent node is computed using the previously computed (vector,matrix) pair (p_1, P_1) as well as (a, A). The vectors are used for classifying each phrase using the same $\mathrm{softmax}$ classifier as in Eq. 1.

4.3 RNTN:Recursive Neural Tensor Network

One problem with the MV-RNN is that the number of parameters becomes very large and depends on the size of the vocabulary. It would be cognitively more plausible if there was a single powerful composition function with a fixed number of parameters. The standard RNN is a good candidate for such a function. However, in the standard RNN, the input vectors only implicitly interact through the nonlinearity (squashing) function. A more direct, possibly multiplicative, interaction would allow the model to have greater interactions between the input vectors.

Motivated by these ideas we ask the question: Can a single, more powerful composition function perform better and compose aggregate meaning from smaller constituents more accurately than many input specific ones? In order to answer this question, we propose a new model called the Recursive Neural Tensor Network (RNTN). The main idea is to use the same, tensor-based composition function for all nodes.

Fig. 5 shows a single tensor layer. We define the output of a tensor product $h \in \mathbb{R}^d$ via the following vectorized notation and the equivalent but more detailed notation for each slice $V^{[i]} \in \mathbb{R}^{d\times d}$:

$$h = \begin{bmatrix} b \\ c \end{bmatrix}^T V^{[1:d]} \begin{bmatrix} b \\ c \end{bmatrix}; h_i = \begin{bmatrix} b \\ c \end{bmatrix}^T V^{[i]} \begin{bmatrix} b \\ c \end{bmatrix}.$$

where $V^{[1:d]} \in \mathbb{R}^{2d\times 2d\times d}$ is the tensor that defines multiple bilinear forms.

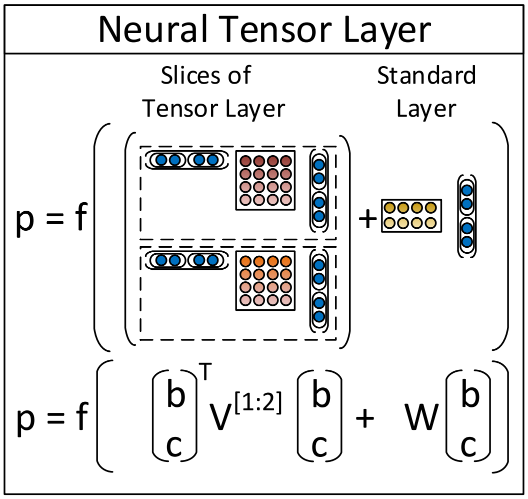


Figure 5: A single layer of the Recursive Neural Tensor Network. Each dashed box represents one of d-many slices and can capture a type of influence a child can have on its parent.

The RNTN uses this definition for computing p_1:

$$p_1 = f\left(\begin{bmatrix} b \\ c \end{bmatrix}^T V^{[1:d]} \begin{bmatrix} b \\ c \end{bmatrix} + W \begin{bmatrix} b \\ c \end{bmatrix}\right),$$

where W is as defined in the previous models. The next parent vector p_2 in the tri-gram will be computed with the same weights:

$$p_2 = f\left(\begin{bmatrix} a \\ p_1 \end{bmatrix}^T V^{[1:d]} \begin{bmatrix} a \\ p_1 \end{bmatrix} + W \begin{bmatrix} a \\ p_1 \end{bmatrix}\right).$$

The main advantage over the previous RNN model, which is a special case of the RNTN when V is set to 0, is that the tensor can directly relate input vectors. Intuitively, we can interpret each slice of the tensor as capturing a specific type of composition.

An alternative to RNTNs would be to make the compositional function more powerful by adding a second neural network layer. However, initial experiments showed that it is hard to optimize this model and vector interactions are still more implicit than in the RNTN.

4.4 Tensor Backprop through Structure

We describe in this section how to train the RNTN model. As mentioned above, each node has a $\mathrm{softmax}$ classifier trained on its vector representation to predict a given ground truth or target vector t. We assume the target distribution vector at each node has a 0-1 encoding. If there are C classes, then it has length C and a 1 at the correct label. All other entries are 0.

We want to maximize the probability of the correct prediction, or minimize the cross-entropy error between the predicted distribution $y^i \in \mathbb{R}^{C\times 1}$ at node i and the target distribution $t^i \in \mathbb{R}^{C\times 1}$ at that node. This is equivalent (up to a constant) to minimizing the KL-divergence between the two distributions. The error as a function of the RNTN parameters $\theta = (V, W, W_s, L)$ for a sentence is:

$$E(\theta) = \sum_i \sum_j t^i_j \log y^i_j + \lambda \|\theta\|^2 \qquad (2)$$

The derivative for the weights of the $\mathrm{softmax}$ classifier are standard and simply sum up from each node's error. We define x^i to be the vector at node i (in the example trigram, the $x^i \in \mathbb{R}^{d\times 1}$'s are (a, b, c, p_1, p_2)). We skip the standard derivative for W_s. Each node backpropagates its error through to the recursively used weights V, W. Let $\delta^{i,s} \in \mathbb{R}^{d\times 1}$ be the $\mathrm{softmax}$ error vector at node i:

$$\delta^{i,s} = \left(W_s^T (y^i - t^i)\right) \otimes f'(x^i),$$

where $\otimes$ is the Hadamard product between the two vectors and f' is the element-wise derivative of f which in the standard case of using $f = \tanh$ can be computed using only $f(x^i)$.

The remaining derivatives can only be computed in a top-down fashion from the top node through the tree and into the leaf nodes. The full derivative for V and W is the sum of the derivatives at each of the nodes. We define the complete incoming error messages for a node i as $\delta^{i,com}$. The top node, in our case p_2, only received errors from the top node's $\mathrm{softmax}$. Hence, $\delta^{p_2,com} = \delta^{p_2,s}$ which we can use to obtain the standard backprop derivative for W (Goller and Küchler, 1996; Socher et al., 2010). For the derivative of each slice $k = 1, \ldots, d$, we get:

$$\frac{\partial E^{p_2}}{\partial V^{[k]}} = \delta^{p_2,com}_k \begin{bmatrix} a \\ p_1 \end{bmatrix} \begin{bmatrix} a \\ p_1 \end{bmatrix}^T,$$

where $\delta^{p_2,com}_k$ is just the k'th element of this vector. Now, we can compute the error message for the two children of p_2:

$$\delta^{p_2,down} = \left(W^T \delta^{p_2,com} + S\right) \otimes f'\left(\begin{bmatrix} a \\ p_1 \end{bmatrix}\right),$$

where we define

$$S = \sum_{k=1}^{d} \delta_k^{p_2,com} \left(V^{[k]} + \left(V^{[k]}\right)^T\right) \begin{bmatrix} a \\ p_1 \end{bmatrix}$$

The children of p_2, will then each take half of this vector and add their own $\operatorname{softmax}$ error message for the complete δ. In particular, we have

$$\delta^{p_1,com} = \delta^{p_1,s} + \delta^{p_2,down}[d+1:2d],$$

where $\delta^{p_2,down}[d+1:2d]$ indicates that p_1 is the right child of p_2 and hence takes the 2nd half of the error, for the final word vector derivative for a, it will be $\delta^{p_2,down}[1:d]$.

The full derivative for slice $V^{[k]}$ for this trigram tree then is the sum at each node:

$$\frac{\partial E}{\partial V^{[k]}} = \frac{E^{p_2}}{\partial V^{[k]}} + \delta_k^{p_1,com} \begin{bmatrix} b \\ c \end{bmatrix} \begin{bmatrix} b \\ c \end{bmatrix}^T,$$

and similarly for W. For this nonconvex optimization we use AdaGrad (Duchi et al., 2011) which converges in less than 3 hours to a local optimum.

5 Experiments

We include two types of analyses. The first type includes several large quantitative evaluations on the test set. The second type focuses on two linguistic phenomena that are important in sentiment.

For all models, we use the dev set and cross-validate over regularization of the weights, word vector size as well as learning rate and minibatch size for AdaGrad. Optimal performance for all models was achieved at word vector sizes between 25 and 35 dimensions and batch sizes between 20 and 30. Performance decreased at larger or smaller vector and batch sizes. This indicates that the RNTN does not outperform the standard RNN due to simply having more parameters. The MV-RNN has orders of magnitudes more parameters than any other model due to the word matrices. The RNTN would usually achieve its best performance on the dev set after training for 3 - 5 hours. Initial experiments showed that the recursive models worked significantly worse (over 5% drop in accuracy) when no nonlinearity was used. We use $f = \tanh$ in all experiments.

Model	Fine-grained		Positive/Negative	
	All	Root	All	Root
NB	67.2	41.0	82.6	81.8
SVM	64.3	40.7	84.6	79.4
BiNB	71.0	41.9	82.7	83.1
VecAvg	73.3	32.7	85.1	80.1
RNN	79.0	43.2	86.1	82.4
MV-RNN	78.7	44.4	86.8	82.9
RNTN	**80.7**	**45.7**	**87.6**	**85.4**

Table 1: Accuracy for fine grained (5-class) and binary predictions at the sentence level (root) and for all nodes.

We compare to commonly used methods that use bag of words features with Naive Bayes and SVMs, as well as Naive Bayes with bag of bigram features. We abbreviate these with NB, SVM and biNB. We also compare to a model that averages neural word vectors and ignores word order (VecAvg).

The sentences in the treebank were split into a train (8544), dev (1101) and test splits (2210) and these splits are made available with the data release. We also analyze performance on only positive and negative sentences, ignoring the neutral class. This filters about 20% of the data with the three sets having 6920/872/1821 sentences.

5.1 Fine-grained Sentiment For All Phrases

The main novel experiment and evaluation metric analyze the accuracy of fine-grained sentiment classification for all phrases. Fig. 2 showed that a fine grained classification into 5 classes is a reasonable approximation to capture most of the data variation.

Fig. 6 shows the result on this new corpus. The RNTN gets the highest performance, followed by the MV-RNN and RNN. The recursive models work very well on shorter phrases, where negation and composition are important, while bag of features baselines perform well only with longer sentences. The RNTN accuracy upper bounds other models at most n-gram lengths.

Table 1 (left) shows the overall accuracy numbers for fine grained prediction at all phrase lengths and full sentences.

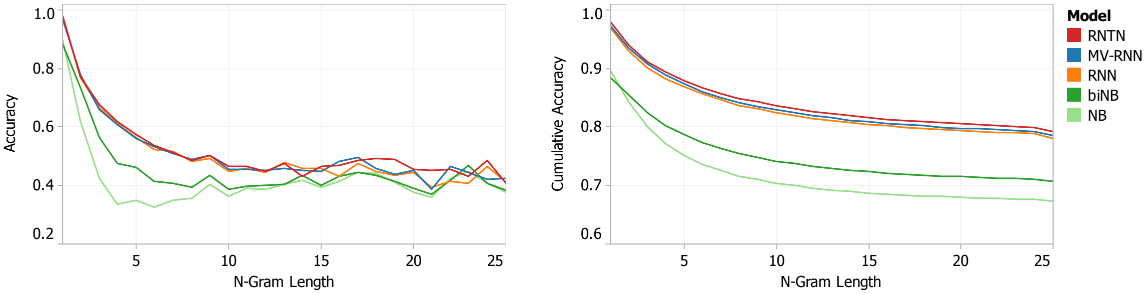


Figure 6: Accuracy curves for fine grained sentiment classification at each n-gram lengths. Left: Accuracy separately for each set of n-grams. Right: Cumulative accuracy of all $\leq n$-grams.

5.2 Full Sentence Binary Sentiment

This setup is comparable to previous work on the original rotten tomatoes dataset which only used full sentence labels and binary classification of positive/negative. Hence, these experiments show the improvement even baseline methods can achieve with the sentiment treebank. Table 1 shows results of this binary classification for both all phrases and for only full sentences. The previous state of the art was below 80% (Socher et al., 2012). With the coarse bag of words annotation for training, many of the more complex phenomena could not be captured, even by more powerful models. The combination of the new sentiment treebank and the RNTN pushes the state of the art on short phrases up to 85.4%.

5.3 Model Analysis: Contrastive Conjunction

In this section, we use a subset of the test set which includes only sentences with an 'X *but* Y' structure: A phrase X being followed by *but* which is followed by a phrase Y. The conjunction is interpreted as an argument for the second conjunct, with the first functioning concessively (Lakoff, 1971; Blakemore, 1989; Merin, 1999). Fig. 7 contains an example. We analyze a strict setting, where X and Y are phrases of different sentiment (including neutral). The example is counted as correct, if the classifications for both phrases X and Y are correct. Furthermore, the lowest node that dominates both of the word *but* and the node that spans Y also have to have the same correct sentiment. For the resulting 131 cases, the RNTN obtains an accuracy of 41% compared to MV-RNN (37), RNN (36) and biNB (27).

5.4 Model Analysis: High Level Negation

We investigate two types of negation. For each type, we use a separate dataset for evaluation.

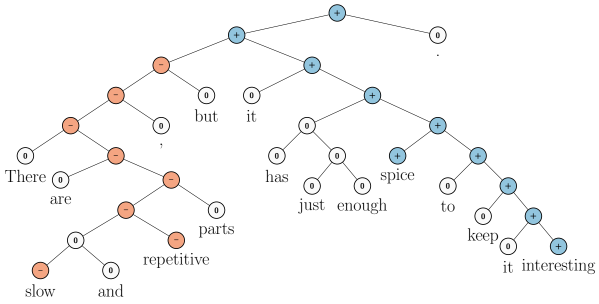


Figure 7: Example of correct prediction for contrastive conjunction X *but* Y.

Set 1: Negating Positive Sentences. The first set contains positive sentences and their negation. In this set, the negation changes the overall sentiment of a sentence from positive to negative. Hence, we compute accuracy in terms of correct sentiment reversal from positive to negative. Fig. 9 shows two examples of positive negation the RNTN correctly classified, even if negation is less obvious in the case of 'least'. Table 2 (left) gives the accuracies over 21 positive sentences and their negation for all models. The RNTN has the highest reversal accuracy, showing its ability to structurally learn negation of positive sentences. But what if the model simply makes phrases very negative when negation is in the sentence? The next experiments show that the model captures more than such a simplistic negation rule.

Set 2: Negating Negative Sentences. The second set contains negative sentences and their negation. When negative sentences are negated, the sentiment treebank shows that overall sentiment should become *less negative*, but not necessarily positive. For instance, 'The movie was terrible' is negative but the 'The movie was not terrible' says only that it was less bad than a terrible one, not that it was good (Horn, 1989; Israel, 2001). Hence, we evaluate ac-

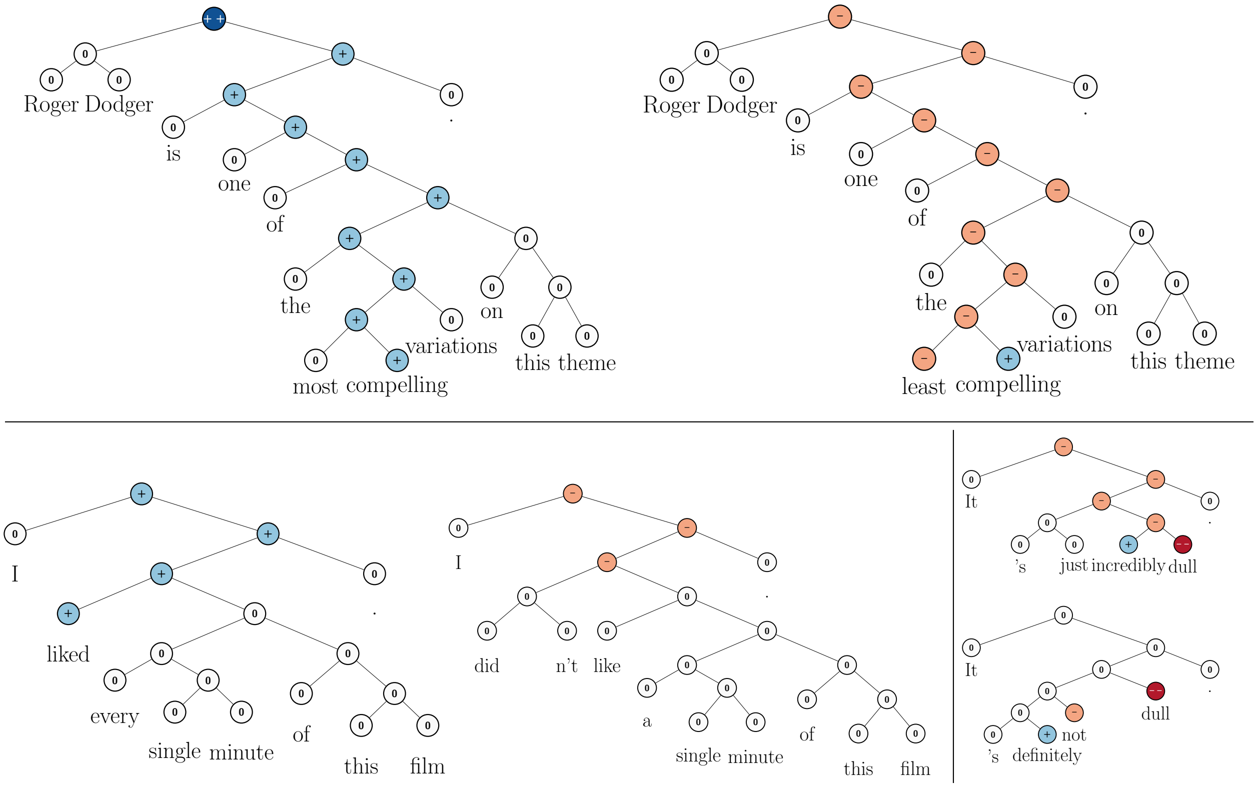


Figure 9: RNTN prediction of positive and negative (bottom right) sentences and their negation.

Model	Accuracy	
	Negated Positive	Negated Negative
biNB	19.0	27.3
RNN	33.3	45.5
MV-RNN	52.4	54.6
RNTN	**71.4**	**81.8**

Table 2: Accuracy of negation detection. Negated positive is measured as correct sentiment inversions. Negated negative is measured as increases in positive activations.

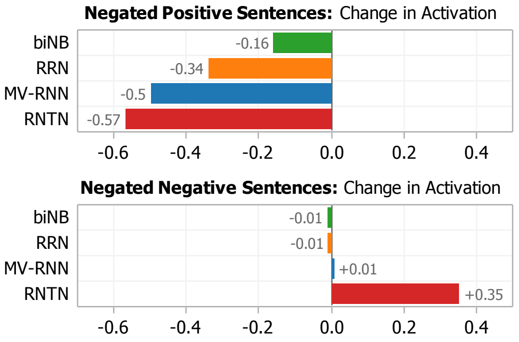


Figure 8: Change in activations for negations. Only the RNTN correctly captures both types. It decreases positive sentiment more when it is negated and learns that negating negative phrases (such as *not terrible*) should increase neutral and positive activations.

curacy in terms of how often each model was able to increase non-negative activation in the sentiment of the sentence. Table 2 (right) shows the accuracy. In over 81% of cases, the RNTN correctly increases the positive activations. Fig. 9 (bottom right) shows a typical case in which sentiment was made more positive by switching the main class from negative to neutral even though both *not* and *dull* were negative. Fig. 8 shows the changes in activation for both sets. Negative values indicate a decrease in average positive activation (for set 1) and positive values mean an increase in average positive activation (set 2). The RNTN has the largest shifts in the correct directions. Therefore we can conclude that the RNTN is best able to identify the effect of negations upon both positive and negative sentiment sentences.

n	Most positive n-grams	Most negative n-grams
1	engaging; best; powerful; love; beautiful	bad; dull; boring; fails; worst; stupid; painfully
2	excellent performances; A masterpiece; masterful film; wonderful movie; marvelous performances	worst movie; very bad; shapeless mess; worst thing; instantly forgettable; complete failure
3	an amazing performance; wonderful all-ages triumph; a wonderful movie; most visually stunning	for worst movie; A lousy movie; a complete failure; most painfully marginal; very bad sign
5	nicely acted and beautifully shot; gorgeous imagery, effective performances; the best of the year; a terrific American sports movie; refreshingly honest and ultimately touching	silliest and most incoherent movie; completely crass and forgettable movie; just another bad movie. A cumbersome and cliche-ridden movie; a humorless, disjointed mess
8	one of the best films of the year; A love for films shines through each frame; created a masterful piece of artistry right here; A masterful film from a master filmmaker,	A trashy, exploitative, thoroughly unpleasant experience ; this sloppy drama is an empty vessel.; quickly drags on becoming boring and predictable.; be the worst special-effects creation of the year

Table 3: Examples of n-grams for which the RNTN predicted the most positive and most negative responses.

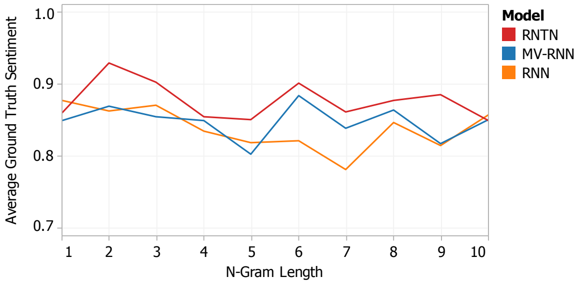


Figure 10: Average ground truth sentiment of top 10 most positive n-grams at various n. The RNTN correctly picks the more negative and positive examples.

5.5 Model Analysis: Most Positive and Negative Phrases

We queried the model for its predictions on what the most positive or negative n-grams are, measured as the highest activation of the most negative and most positive classes. Table 3 shows some phrases from the dev set which the RNTN selected for their strongest sentiment.

Due to lack of space we cannot compare top phrases of the other models but Fig. 10 shows that the RNTN selects more strongly positive phrases at most n-gram lengths compared to other models.

For this and the previous experiment, please find additional examples and descriptions in the supplementary material.

6 Conclusion

We introduced Recursive Neural Tensor Networks and the Stanford Sentiment Treebank. The combination of new model and data results in a system for single sentence sentiment detection that pushes state of the art by 5.4% for positive/negative sentence classification. Apart from this standard setting, the dataset also poses important new challenges and allows for new evaluation metrics. For instance, the RNTN obtains 80.7% accuracy on fine-grained sentiment prediction across all phrases and captures negation of different sentiments and scope more accurately than previous models.

Acknowledgments

We thank Rukmani Ravisundaram and Tayyab Tariq for the first version of the online demo. Richard is partly supported by a Microsoft Research PhD fellowship. The authors gratefully acknowledge the support of the Defense Advanced Research Projects Agency (DARPA) Deep Exploration and Filtering of Text (DEFT) Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-13-2-0040, the DARPA Deep Learning program under contract number FA8650-10-C-7020 and NSF IIS-1159679. 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.

Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3, March.

D. Blakemore. 1989. Denial and contrast: A relevance theoretic analysis of 'but'. *Linguistics and Philosophy*, 12:15–37.

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

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. Duchi, E. Hazan, and Y. Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12, July.

K. Erk and S. Padó. 2008. A structured vector space model for word meaning in context. In *EMNLP*.

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*.

E. Grefenstette, G. Dinu, Y.-Z. Zhang, M. Sadrzadeh, and M. Baroni. 2013. Multi-step regression learning for compositional distributional semantics. In *IWCS*.

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

L. R. Horn. 1989. *A natural history of negation*, volume 960. University of Chicago Press Chicago.

E. H. Huang, R. Socher, C. D. Manning, and A. Y. Ng. 2012. Improving Word Representations via Global Context and Multiple Word Prototypes. In *ACL*.

M. Israel. 2001. Minimizers, maximizers, and the rhetoric of scalar reasoning. *Journal of Semantics*, 18(4):297–331.

R. Jenatton, N. Le Roux, A. Bordes, and G. Obozinski. 2012. A latent factor model for highly multi-relational data. In *NIPS*.

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

R. Lakoff. 1971. If's, and's, and but's about conjunction. In Charles J. Fillmore and D. Terence Langendoen, editors, *Studies in Linguistic Semantics*, pages 114–149. Holt, Rinehart, and Winston, New York.

A. Merin. 1999. Information, relevance, and social decisionmaking: Some principles and results of decision-theoretic semantics. In Lawrence S. Moss, Jonathan Ginzburg, and Maarten de Rijke, editors, *Logic, Language, and Information*, volume 2. CSLI, Stanford, CA.

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

K. Moilanen and S. Pulman. 2007. Sentiment composition. In *In Proceedings of Recent Advances in Natural Language Processing*.

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

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.

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

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

L. Polanyi and A. Zaenen. 2006. Contextual valence shifters. In W. Bruce Croft, James Shanahan, Yan Qu, and Janyce Wiebe, editors, *Computing Attitude and Affect in Text: Theory and Applications*, volume 20 of *The Information Retrieval Series*, chapter 1.

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

M. Ranzato and A. Krizhevsky G. E. Hinton. 2010. Factored 3-Way Restricted Boltzmann Machines For Modeling Natural Images. *AISTATS*.

V. Rentoumi, S. Petrakis, M. Klenner, G. A. Vouros, and V. Karkaletsis. 2010. United we stand: Improving sentiment analysis by joining machine learning and rule based methods. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta.

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

B. Snyder and R. Barzilay. 2007. Multiple aspect ranking using the Good Grief algorithm. In *HLT-NAACL*.

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, C. Lin, A. Y. Ng, and C.D. Manning. 2011a. 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. 2011b. Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. In *EMNLP*.

R. Socher, B. Huval, C. D. Manning, and A. Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *EMNLP*.

I. Sutskever, R. Salakhutdinov, and J. B. Tenenbaum. 2009. Modelling relational data using Bayesian clustered tensor factorization. In *NIPS*.

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

H. Wang, D. Can, A. Kazemzadeh, F. Bar, and S. Narayanan. 2012. A system for real-time twitter sentiment analysis of 2012 u.s. presidential election cycle. In *Proceedings of the ACL 2012 System Demonstrations*.

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*.

D. Yu, L. Deng, and F. Seide. 2012. Large vocabulary speech recognition using deep tensor neural networks. In *INTERSPEECH*.

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

L. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*.