Sentiment Analysis with Deeply Learned Distributed Representations of Variable Length Texts

James Hong
Stanford University
jamesh93@stanford.edu

Michael Fang
Stanford University
mjfang@stanford.edu

Abstract

Learning good semantic vector representations for phrases, sentences and paragraphs is a challenging and ongoing area of research in natural language processing and understanding. In this project, we survey and implement several deep-learning and deep-learning-inspired approaches and evaluate these algorithms on several sentiment-labeled datasets and analysis tasks. In doing so, we demonstrate new state-of-the-art performance on the IMDB Large Movie Review Dataset [5] using highly-tuned paragraph vectors [4], and highly competitive performance on the Stanford Sentiment Treebank dataset [8] using Deep Recursive-NNs and LSTMs for both binary and fine classification tasks. Finally, we compare and analyze each model's performance on our selection of sentiment analysis tasks.

1 Introduction

The problem of creating represention of text for computational analysis is core to the field of NLP. In this project, we focus on looking at various models to create vector representations for the problem of sentiment analysis, or, more specifically, predicting the positive or negative polarity of segments of text.

2 Background/Related Work

Initial attempts to represent text used one-hot vectors to represent a word, where each dimension of the vector corresponds to a distinct word. And in the simple "bag of words" representation of a document, the document is represented as the sum of the one-hot vectors of the words. However, these representations lose much of the semantic meaning that we associate with text.

More recent work has focused on creating vector representations for both words and documents, with the idea of retaining as much information as possible. In word2vec [6], vector representations are computed for each word, with the result being that words whose meanings are related are generally closer in terms of euclidean distance than between unrelated words.

Deep learning models have also been effective in tackling this problem. Recurrent neural networks, by being able to retain memory between training examples, allows it to capture relations between words, within models such as LSTMs having the ability to remember important information across long stretches of time [1]. In this project, we attempt to replicate some of these results.

3 Approach

3.1 Datasets

We evaluate the performance of our models on a number of datasets, including the Stanford Large Movie Review Dataset (IMDB) dataset [5] and Stanford Sentiment Treebank dataset [8] derived from Rotten Tomatoes movie reviews [7]. These two datasets allow us to evaluate and compare performance against existing publications on both binary and fine-grained classification tasks.

3.2 Models

We evaluate the following models and algorithms:

3.2.1 Average of Word Vectors

We train word vectors in an unsupervised manner on the training set. To generate sentence or phrase representations, we average the vectors for words contained in a text. This has the advantage of being simple and fast to train once the word embeddings have been trained. A disadvantage is that it loses information about word ordering. Some additional variants on this approach include using a weighted average based on the tf-idf transform.

3.2.2 Paragraph Vector

Paragraph vectors is a model for learning representations for documents that learns in the same vein as learning word vectors. Training word vectors occurs as normal, except that an additional vector representing the paragraph is added to the task whenever the sampled window comes from that paragraph. Thus, as more samples are taken over time from the paragraph, errors are backpropagated into the vector. This works for both CBOW and Skip-gram architectures. This has been shown to work better than just averaging the word vectors together [4].

For the sentiment analysis task, we learn paragraph vector representations on both training and test data. Then, the paragraph vectors of the training set are used as features train a logistic classifier or a multi-layer perceptron to predict the overall sentiment (positive or negative) of the paragraph. We tune hyper-parameters using 5-fold cross validation or a hold out validation set, respectively for the logistic regression classifier and MLP. Finally, we measure model accuracy on the unseen test set.

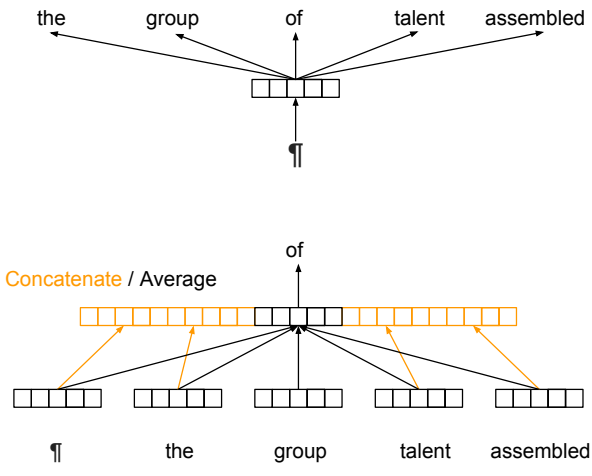

Figure 1: Paragraph vectors are trained to either (above) predict words in a sampled window from within the paragraph as in Skip-gram, or appended alongside context words to predict a center word from a sampled window, as in CBOW.

In Autumn quarter, we successfully replicated and exceeded the previous state-of-the-art performance reported by Le and Mikolov on the IMDB dataset using paragraph vectors [4]. Using knowledge obtained from CS224D, we trained deeper models for the final classification stage on the IMDB task and achieved an error rate of 5.496%, a 0.314% reduction in error over our previous CS224N result [2] and a 1.924% reduction from the results attained by Le and Mikolov [4]. Moreover, we have greatly reduced the training time and improved the consistency of our MLP implementation on the IMDB data. Specifically, for CS224D, we also extend our work to the Sentiment Treebank results also reported by Le and Mikolov. In our experience, paragraph vectors perform sub-optimally compared to those by Le and Mikolov on the Stanford Sentiment Treebank dataset, and in the following sections, we provide some error analysis and insights as to why this is the case.

3.2.3 Long Short-Term Memory (LSTM)

Long short-term memory models are a type of recurrent neural network. In recurrent neural networks (RNN), predictions are made sequentially, and the hidden layer from one prediction is fed to the hidden layer of the next prediction. This gives the network "memory", in the sense that the results from previous predictions can inform future predictions. LSTMs add additional factors to a traditional RNN that give it more of a fine-grained control over memory. These factors control 1) how much the current input matters in creating the new memory, 2) how much the previous memories matters in creating the new memory, and 3) what parts of the memory are important in generating the output.

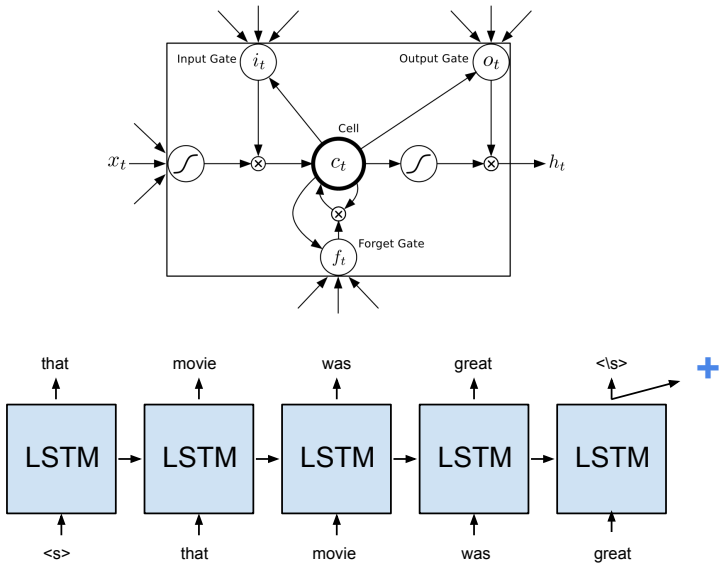

Figure 2: This diagram shows the setup for LSTM using only the last hidden dimension for sentiment prediction. The LSTM is jointly trained for language modeling. Alternate variants that we experimented included averaging hidden states and only softmax/cross-entropy loss associated with sentiment analysis.

We initially trained the LSTM for sentiment analysis by using only the final hidden state of the top layer of the LSTM as input to a softmax or logistic classifier, as in [9]. However, we found that by instead averaging the hidden states over the entirety of the sentence, we were able to get better results. One explanation would be that doing so increases the level of feedback, particularly to earlier words. The LSTM was trained using rmsprop, which worked better than using an annealed learning rate. Dropout was found to be a powerful regularizer, even when the network was only one layer deep. Increasing the number of layers, where the output of one layer is fed as input to the next, was found to sometimes be effective. We found that initializing using GloVe vectors for

the word embeddings substantially improved LSTM performance, at least for Treebank dataset. Our experiments showed that L2 regularization was not particularly helpful.

3.2.4 Deep Recursive-NN

We implemented a deep recursive neural network. Like the shallow RNN, we back-propagate error though a tree structure. However, we add an additional hidden layer to the tree structured back-propagation. Our implementation is similar to the DRNN presented by Irsoy and Cardle [3], except that for leaf nodes we have 2 two sets of word-vectors. We use ReLU activation units for non-linearity.

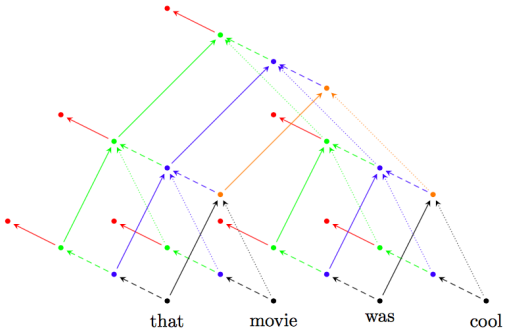

Figure 3: An example of a 3-layer deep recursive neural network from Irsoy and Cardie [3]. For our experiments, we trained a 2-layer deep RNN. We also added an additional 3rd, untied hidden layer, between the 2nd layer and the softmax.

We tested our deep recursive-NN on the Stanford Sentiment Treebank data.

4 IMDB Dataset

The Stanford Large Movie Review (IMDB) Dataset [5] consists of 50,000 "highly polar", binary labeled reviews from IMDB. These reviews are split 50:50 into training and testing sets. The distribution of labels within each subset of data is balanced. The dataset also includes a further 50,000 unlabeled reviews which may be used for unsupervised training. We found that reviews averaged 267.9 tokens in length with a standard deviation of 198.8 tokens; the precise distribution of review lengths is shown below.

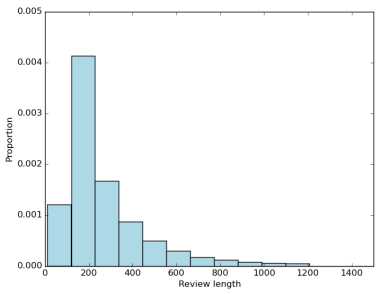

Figure 4: Distribution of review lengths in the IMDB dataset.

4.1 Model Tuning

4.1.1 Paragraph Vector

In training our paragraph vectors on the IMDB binary classification task, we found the optimal hyperparameters for the word and paragraph vector training to be 150 dimensions, 25 training epochs, an initial learning rate of 0.05, negative sampling loss with 25 negative samples, and vocabulary downsampling of 1e-4 and 1e-2 for CBOW and Skip-Gram respectively. Specifically, we found that by tuning the downsampling parameter separately for the two models and then concatenating the resulting paragraph vectors, we were able to obtain significantly better performance than either model alone; we believe, that the two models are able capture different, orthogonal aspects of the data.

4.1.2 LSTM

On the IMDB dataset, our one-layer LSTM performed best with word vector and hidden dimensions of 100. However, we tested dimensions ranging from 50 to 200; overall, the LSTM easily attains high accuracy on the training set while failing to generalize to the test set. Optimal performance was obtained after 10-17 training epochs.

4.2 Results

Model	Train	Test
NBSVM-bi (Wang & Manning, 2012)		0.912
Paragraph Vector (Le and Mikolov. 2014)		0.927
Averaged Word Vector	0.896	0.883
Paragraph Vector (LogReg)	0.975	0.944
Paragraph Vector (2-layer MLP)	0.971	**0.945**
LSTM	0.983	0.891

Table 1: Results on IMDB dataset binary classification task.

4.3 Analysis

Our Paragraph Vector implementation worked surprisingly well on the IMDB dataset, with performance exceeding that of the original paper. Perhaps using different downsampling parameters for the different architectures may have perhaps captured different information about the data that the classifiers could capitalize on. The LSTM did not seem to perform as well on this task, though.

5 Stanford Sentiment Treebank Dataset

The Stanford Sentiment Treebank Dataset consists of 11,855 reviews from Rotten Tomatoes. They are split across train, dev and test sets, containing 8,544, 1,101, and 2,210 reviews respectively. Reviews are labeled on a 5 point scale corresponding to very negative, negative, neutral, positive, and very positive. In accordance with existing results published on this dataset [8, 4], we present our final accuracy and error scores on both binary and fine classification tasks over entire reviews. The average review length is 20.1 tokens with a standard deviation of 9.3.

5.1 Model Tuning

5.1.1 Paragraph Vectors

We were ultimately unable to reproduce Le and Mikolov's paragraph vector performance on the sentiment treebank dataset. The best hyper-parameters we found for training the word and paragraph vectors was a word and paragraph vector dimension of 400, negative sampling loss with 5 negative samples, 25 epochs for training, initial learning rate of 0.025 and vocab sub-sampling rates of 5e-3 and 1e-1 for CBOW and Skip-Gram respectively. CBOW and Skip-Gram vectors were concatenated for better performance. We also experimented with the hierarchical-softmax loss, but found that this performed much worse due to overfitting to the training set, which is used to initialize the vocabulary.

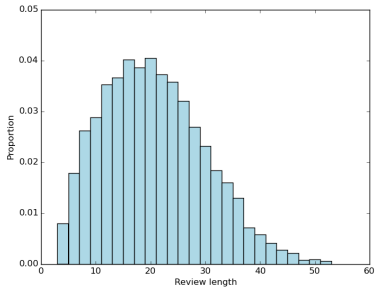

Figure 5: Distribution of review lengths in the Stanford Sentiment Treebank dataset.

5.1.2 LSTM

For our best results using LSTM on the Stanford Sentiment Treebank Dataset, we used an LSTM 3 layers deep, with word and hidden vector dimensions of 100, and dropout regularization with a drop rate of 0.5. Updates were applied using RMSProp with an initial learning rate of 1, decay of 0.99 and epsilon of 1e-8.

5.2 Deep Recursive-NN

For our deep RNN, we used word vector and hidden dimensions of 75 (100 when pretrained Glove vectors were used). The initial learning rate was set to 0.07 and adagrad was used for updates. We set L2 regularization to 0.01 and applied a dropout regularization rate of 0.9 to each hidden layer. Models were trained for up to 100 epochs. Generally, optimal performance is reached at around 40-60 epochs.

5.3 Results

Model	Train	Test
Averaged Word Vectors	0.806	0.786
Paragraph Vector	0.791	0.764
LSTM	0.935	0.843
Deep Recursive-NN	0.938	0.847

Table 2: Accuracy on Stanford Sentiment Treebank dataset for binary classification at the root.

Model	Train	Test
Averaged Word Vectors	0.549	0.406
Paragraph Vector	0.397	0.379
Paragraph Vector (pretrained word-embeddings)	0.411	0.391
LSTM	0.747	0.426
LSTM (w/ Glove word-vectors)	0.560	0.456
Deep Recursive-NN	0.584	0.469

Table 3: Accuracy on Stanford Sentiment Treebank dataset for 5-way classification at the root.

Model	Fine (5-class)	Binary
DCNN (Blunsom, et al. 2014)	0.485	0.868
RNTN (Socher, et al. 2013)	0.457	0.854
CNN-non-static (Kim, 2014)	0.480	0.872
CNN-multi-channel (Kim, 2014)	0.474	0.881
DRNN w. pretrained word-embeddings (Irsoy and Cardie, 2014)	0.498	0.866
Paragraph Vector (Le and Mikolov. 2014)	0.487	0.878
Dependency Tree-LSTM (Tai, et al, 2015)	0.484	0.857
Constituency Tree-LSTM (Tai, et al, 2015)	0.439	0.820
Constituency Tree-LSTM (Glove vectors) (Tai, et al, 2015)	0.510	0.880
Paragraph Vector	0.391	0.798
LSTM	0.456	0.843
Deep Recursive-NN	0.469	0.847

Table 4: Our test accuracy on Stanford Sentiment Treebank dataset for binary classification at the root compared to current state-of-the-art.

Model	Train	Test
RNTN (Socher, et al. 2013)		0.807
Deep-RNN	0.876	0.807

Table 5: Accuracy on Stanford Sentiment Treebank dataset for 5-way classification at all nodes.

5.4 Analysis

Overall, both our LSTM and Deep Recursive-NN performed very competitively and came close to matching other state-of-the-art algorithms on both binary and fine-grained sentiment analysis on the Stanford Sentiment Treebank dataset. Our best performance was obtained using the Deep Recursive-NN with fine and binary classification accuracies 46.9% and 84.7% respectively. These results are only exceeded by convolutional neural nets, Le's reported performance with paragraph vectors, and dependency-tree LSTMs when pretrained word-embeddings are removed from consideration.

5.4.1 Paragraph Vector

We were unable to replicate Le and Mikolov's results for paragraph vectors on the Sentiment Treebank Dataset, attaining accuracies of 0.391 and 0.798 for fine and binary classification respectively, compared to 0.487 and 0.878 [4]. To identify the causes of these discrepancies, we examined both the paragraph and word vectors produced by our extended version of word2vec. In doing so, we found that word embeddings trained on only the Sentiment Treebank dataset alone were of significantly lower quality than those trained on the much larger IMDB dataset. Using our pre-trained word embeddings from the IMDB dataset helped boost accuracy from 0.379 to 0.391 on the test set. A small vocabulary size of roughly 4.5 thousand words was another consequence of the much smaller dataset.

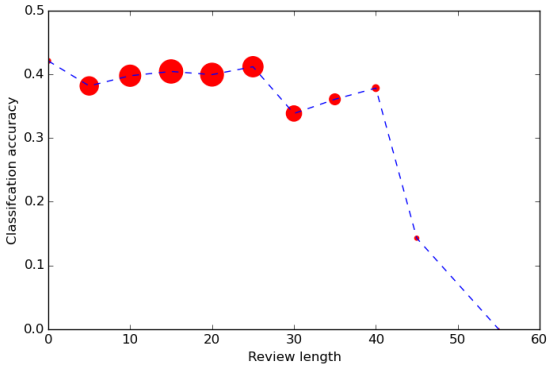

Figure 6: Classification accuracy vs. review length in tokens. Initially we suspected that longer sentences would be easier to classify and learn paragraph vectors over. This turned out not to be the case. Instead, upon further analysis, we found that word2vec and our derived paragraph2vec generates poor quality word vectors that impede performance across the board. We verified this using the word distance metric code provided by word2vec [6].

5.4.2 LSTM

Our LSTM implementation was not as effective as the results reported in [9]. One reason could be that we did not start using pre-initialized GloVe vectors until relatively late in our parameter tuning. Our experiments showed that increasing the dimensions (using sizes 150, 200, and 300) of the LSTM were not effective, but this was with randomly initialized word embeddings. Perhaps using 300 dimensional GloVe vectors would indeed have been more effective. Furthermore, it is worth

5.4.3 Deep Recursive-NN

Our implementation of DRNN matches other well performing architectures such RNTN in performance on both fine and binary classification. In their original paper, Irsoy and Cardie report an accuracies of 49.8% and 86.6% for fine and binary respectively [3]. However, those results were obtained with pretrained word embeddings on the Google News Dataset with (1̃00B words) [3]. Thus, we feel that those results are not directly comparable with ours or RNTN results reported by Socher, et al. in 2013 [8]. Similarly, our DRNN outperforms the Constituency Tree-LSTM model proposed by Tai, et al. in 2015 [9] when word vectors are initialized randomly. The results reported above for DRNN are obtained with randomly initialized word vectors. We have tested our DRNN with 100-dimensional Glove vectors, with limited benefits so far. We consider retraining with 300-dimensional Glove vectors and for longer periods to be future work.

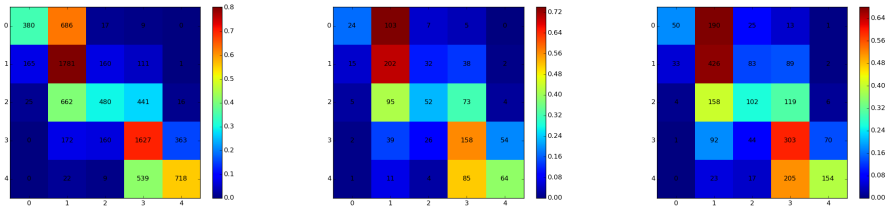

Figure 7: Confusion matrices for fine-grained whole sentence classifcation for Deep Recursive-NN. From left to right: train, dev, test.

6 Conclusion

Ultimately, we have surveyed and implemented a number of architectural and algorithmic approaches to learning distributed vector representations for paragraphs, sentences, and phrases with the ultimate goal of sentiment analysis. We demonstrate new state-of-the-art performance using paragraph vectors on the IMDB dataset. However, paragraph vectors live up to their reputation of being extremely difficult to tune for the second Stanford Sentiment Treebank Dataset. For fine and binary classification on this latter dataset, we demonstrate competitive performance using LSTM and Deep Recursive-NN models.

7 Future Work

There are a number of ways in which this project could be expanded. One direction is using increasingly complicated models. We were not quite able to finish an implementation of a bidirectional LSTM, which has been shown by [9] to be effective as well for sentiment analysis, being able to capture information at every time step from both previous and future text.

Another direction would be to apply our models onto other sorts of sentiment tasks. We would like to see if such models can capture tricky implicit information of human communication such as humor or sarcasm.

8 Acknowledgment

Our deepest thanks go to Thang Minh Luong, for allowing us to build off of his LSTM codebase and for giving valuable advice.

References

[1] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[2] James Hong, Michael Fang, and Vivek Jain. Learning distributed representations for variable length texts. dec 2014.

[3] Ozan Irsoy and Claire Cardie. Deep recursive neural networks for compositionality in language. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2096–2104. Curran Associates, Inc., 2014.

[4] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053, 2014.

[5] Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 142–150. Association for Computational Linguistics, 2011.

[6] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[7] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 115–124. Association for Computational Linguistics, 2005.

[8] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer, 2013.

[9] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. *CoRR*, abs/1503.00075, 2015.