Predicting Sentiment from Rotten Tomatoes Movie Reviews

Jean Y. Wu (jeaneis@stanford.edu)
Symbolic Systems, Stanford University

Yuanyuan Pao (ypao@stanford.edu)
Electrical Engineering, Stanford University

Abstract

The aim of the project is to experiment with different machine learning algorithms to predict the sentiment of unseen reviews from the improved corpus that has the additional sentiment information of all sub-phrases. We use the Rotten Tomatoes movie review corpus[1] that has been greatly improved upon, and annotated with a fine sentiment score via Amazon Mechanical Turk. We want to see whether having a finer sentiment annotations for every span of a parsed sentence in the training set would help improve the accuracy in predicting the overall sentiment of unseen sentences.

Keywords: Sentiment analysis, Naive Bayes, Support Vector Machine, Movie Reviews, Natural Language Processing, Machine Learning, Deep Learning, Neural Network

Introduction

With the increasing ubiquity of social media such as Twitter and review sites like Yelp and Rotten Tomatoes, it is important to be able to automatically make sense of these large amounts of subjective opinionated data. Sentiment analysis, using natural language processing and machine learning techniques to characterize subjective human opinions or sentiments, has been rapidly gaining popularity as a method of analyzing these large corpora for such diverse applications such as predicting trends in the stock market, and characterizing diurnal and seasonal moods such as seasonal affective disorder (Golder & Macy, 2011).

Most of the work to date (Wang & Manning, 2012) has been identifying how the presence of individual words in an excerpt, such as a tweet or movie review, contributes to the sentiment of the entire excerpt (a so-called "bag-of-words" model). In this work, we collect subjective human sentiments of subspans of excerpts, down to the sentiments of individual words. Incorporating the sentiment of individual words (and progressively longer sub-phrases and phrases) into learning algorithms would give us more insight into how the sentiment of the entire excerpt is formed from its constituents. Our hypothesis is that incorporating finer-grained sentiment analysis (i.e., involving the sentiment of the constituent sub-phrases) would lead to an improved accuracy on novel examples. We test and report the accuracy of various machine learning techniques in predicting the sentiment of movie ratings from the sentiments of the constituent sub-phrases. We also discuss the possibility of building a database of sentiments of common words and sub-phrases that would allow algorithms to estimate the same fine-grained sentiment on novel examples without having to laboriously collect fine-grained subjective ratings, which would allow the results presented here to be generalized to other data corpora.

Within the past decade, deep learning has become increasingly popular as a technique for modeling various behaviors while identifying the best features to use as inputs for the resulting model. For our data, we want to input only the n-grams that are identified by the parse trees and use their word representation vectors to predict their sentiments. We then tune the model parameters to identify the model with the best validation accuracy and then report the final model's performance on the test set as well as its precision and recall for each of the ratings from 1-star to 5-stars.

[1] Sentence Polarity Dataset v1.0: http://www.cs.cornell.edu/people/pabo/movie-review-data/

Movie Review Corpus

We perform our tasks on an improved version of an existing movie review dataset and compare with published results of the original dataset.

Sentence Polarity Dataset We consider the corpus of movie review excerpts from the Rotten Tomatoes (RT) website, which was originally collected and published by (Pang & Lee, 2005). To obtain our version of dataset with an improved quality, we first cleaned up the the dataset by removing all the residual HTML tags and restoring the original capitalization of the words. The cleansed sentences were then parsed into a binarized tree using the Stanford Parser (Klein & Manning, 2003). We proceeded to collect sentiment ratings for every span of these binarized trees.

Sentiment Rating Collection We collected subjective human sentiment ratings thru the Amazon Mechanical Turk (AMT) platform. The phrase associated with each span of the parsed trees was given to at least 3 different human judges to rate a sentiment score on the scale of 1 (most negative) to 25 (most positive). The sentiment rating for that phrase would then be averaged, and normalized to 0 to 1, to replace the Rotten Tomatoes' boolean sentence polarity as the ground truth.

Dataset Statistics We split our dataset into a ratio of 7:1:2 for training, cross-validation and test sets, which we use for the most part of this paper. Later on, we would also perform our tasks on 10-fold cross-validation for comparison with other published results.

Here we show an analysis on the standard split sets to ensure that our test results are not skewed due to the imbalance of positive and negative phrases among each set.

We show the proportion of the data that are rated as exactly neutral (averaged score of 0.5) in the dataset, which would be rounded up to a positive score for boolean prediction tasks.

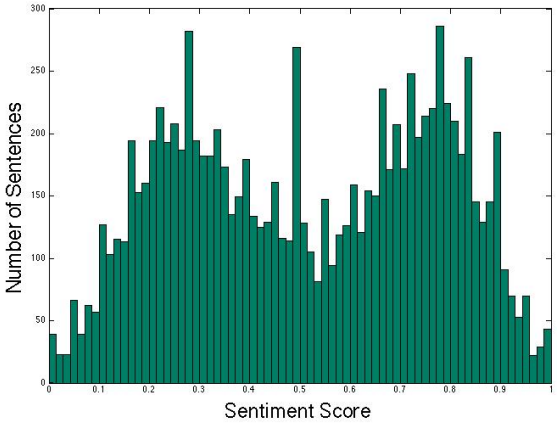

Figure 1: Distribution of the normalized AMT sentiment ratings of the sentence-level nodes. Notice the bimodal shape of the distribution, which indicates that most sentences would either be positive or negative.

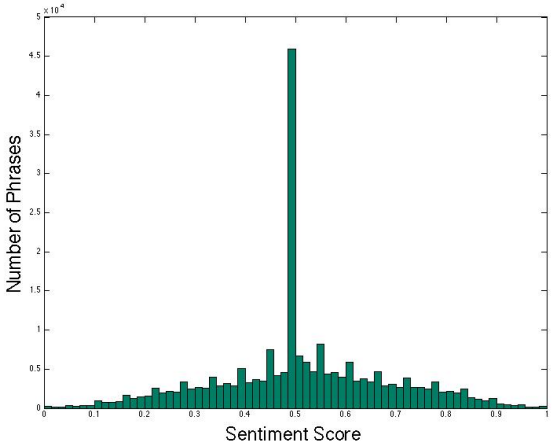

Figure 2: Distribution of the normalized AMT sentiment ratings across all phrases in the dataset. Notice the peak in the neutral score, which indicates a high volume of phrases with no strong polarity sentiment.

The complete dataset is composed of 10215 sentences or 211245 unique phrases (*all-spans*), 21437 of which are unigrams ($|V|_{unigram} = 21437, |V|_{unigram \cup bigram} = 135248$).

The training set contains 7215 sentences, which has 153969 unique phrases.

The cross-validation set contains 1000 sentences, with equal pos/neg split based on RT labeling, and contains 25284 unique phrases.

Finally, the test set contains 2000 sentences, with equal pos/neg split based on RT labeling. There are 47483 unique phrases ($all - spans$) in this set.

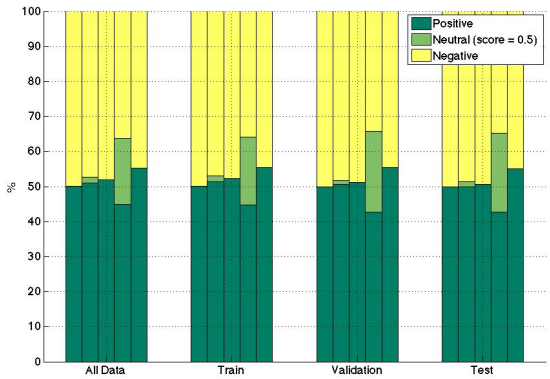

Figure 3: These bars capture the proportion of the positive/(neutral)/negative ratings in the dataset. Bars 1-3 of each group show the proportion of the sentence-level pos/neg ratings, while bars 4-5 of each group show the pos/neg ratings for *all-span*. The first bars in each group are based on the original RT labeling, while the rest are based on the rounded AMT labeling. The second and the fourth bars in each group also highlight the portion of the data that is rated as exactly neutral (averaged score = 0.5), which in our experiments would be rounded up as positive. The third and the fifth bars in each group show the new pos/neg split if we removed all the neutral data from its corresponding set.

Methods

In the rest of this paper, we shall refer to the sentiment of the entire excerpt as the *root* sentiment, and to the sentiments of the subspans as *all-spans*. We are interested to see if

1. training a learning algorithm on *all-spans* would improve prediction on *roots*, or entire novel excerpts; and if
2. training a learning algorithm on *all-spans* would improve predictions on *all-spans*, or the sentiments of every subspan of novel excerpts.

The algorithms that we trained and tested are covered below.

Multinomial Naive Bayes (MNB) We implemented an MNB-Event model on MATLAB to train and test our data.

Support Vector Machine (SVM) We used the LIBLINEAR package (Fan, Chang, Hsieh, Wang, & Lin, 2008) in Matlab to train and test our SVMs.

Deep Learning Our neural network consists of three layers - the input layer, the hidden layer, and the output layer. For our input layer, we feed in the word vector representations for various n-grams. Instead of using a constant length window, we have various length n-grams ranging from one word unigrams to the longest sentences in our training set. As mentioned previously, only the n-grams formed in the parse trees are used as word vectors in training our model so that we are only learning from the n-grams that contribute to the structure of the sentence.. Our hidden layer takes the word vector representation and converts it into a vector of dimension H, another parameter of our neural network, by feeding it through a hyperbolic tangent function with a weight matrix W and bias $b^{(1)}$. For our output layer, we have a logistic regression classifier. So our neural network for one word vector can be represented by the following function:

$$h_\theta(x^{(i)}) = g(U^T f(Wx^{(i)} + b^{(1)}) + b^{(2)}) \quad (1)$$

where f is tanh and g is the sigmoid function.

Because we have a logistic regression classifier, our cost function takes the form of the logistic regression cost function with regularization terms to prevent overfitting. In order to maximize the log likelihood or our parameters, we use LBFGS minimization (of the negative log likelihood) with the following gradients for our parameters:

$$\begin{aligned}
a &= Wx + b^{(1)} && (2)\\
z &= U^T f(a) + b^{(2)} && (3)\\
\frac{\partial J}{\partial U_i} &= \frac{1}{m}\Sigma_{i=1}^{m}(g(z) - y^{(i)}) f(a) + CU_i && (4)\\
\frac{\partial J}{\partial b^{(2)}} &= \frac{1}{m}\Sigma_{i=1}^{m} g(z) - y^{(i)} && (5)\\
\frac{\partial J}{\partial b_k^{(1)}} &= \frac{1}{m}\Sigma_{i=1}^{m}(g(z) - y^{(i)}) U_k f_k'(a) && (6)\\
\frac{\partial J}{\partial W_{kj}} &= \frac{1}{m}\Sigma_{i=1}^{m}(g(z) - y^{(i)}) U_k f_k'(a) x_j^{(i)} + CW_{kj} && (7)\\
\frac{\partial J}{\partial x_j^{(i)}} &= \frac{1}{m}\Sigma_{i=1}^{H}(g(z) - y^{(i)}) U_i f_i'(a) W_{ij}^{(i)} && (8)
\end{aligned}$$

Because the length of our word vector representation x varies with n, the window length, we have a different W matrix and bias $b^{(1)}$ for each value of n. Since U and $b^{(2)}$ only interact with the hidden layer, their dimensions remain the same as n changes; therefore, we can choose to use the same values for all n-grams or to separate them. In tuning our model, we considered both options — with separate U and $b^{(2)}$, we could learn the neural network for each n with the data separated by n-grams, but with the same U and $b^{(2)}$, we have to run through every training n-gram in order to arrive at the right values.

The output layer gives us the probabilities for the sentiment of each n-gram. In order to determine the rating, we bin the output into equal intervals determined by the number of classes we are trying to distinguish, in the case, five ratings. In another approach, we use intervals that are scaled by the observed probabilities of each rating from the training set because the ratings seen during training will affect with scale along which our model will score.

Results

Our MNB and SVM results are reported in Table 1. First, we trained and cross-validated our algorithms on the original (root-only) ratings collected from RT. It is worth noting that the root rating in the RT dataset is awarded by the same person who wrote the review (the excerpt in the RT corpus contains only a short subset of the review). We also tested the accuracy on external root ratings of the excerpts (collected from from AMT), and we see that the accuracy is slightly lower than the RT dataset, which could be due to slight differences in the RT and AMT ratings, especially for more neutral reviews (e.g. a rating of 0.49 vs 0.51 on a scale from 0-1).

Next, we see that training the algorithm on *all-spans* improves the prediction accuracy of root sentiments as compared to the AMT root-root baseline, for both MNB (77.1% vs. 74.6%) and SVM algorithms (76.3% vs. 72.5%). The prediction accuracy on all-spans is slightly lower (MNB: 75.2%, SVM: 72.3%), possibly due to the greater error in predicting the sentiment of all sub-spans of the cross-validation examples.

It's worth nothing that our model tends to predict more positive than negative. This is because in the first setup, we rounded the AMT ratings to 0 and 1, which rendered an imbalance of the postive/negative polarity in the dataset (around 20% of the neutral data were rounded up to positive.) Figure 3 provides a visualization of the breakdown of all the dataset. When we trained and tested with our second setup, which removed all neutral ratings, we see the prediction results to be much more balanced[2].

Optimizing the SVM We optimized the SVM models by trying different regularization parameters, in Fig. 4. In order to compare with existing studies (Wang & Manning, 2012), we trained and tested on the root data from the original RT dataset. For root-training/root-validation on the RT dataset, the best model had a regularization parameter $C = 0.09$ using L2-regularization with L2-loss SVC (primal). With these parameters, the percentage of True Positives (TP) was 37.80%, False Positives (FP) was 13.80%, False Negatives (FN) was 12.20%, and True Negatives (TN) was 36.20%. This leads to an accuracy of 74%, precision (P) of 73.3% and recall (R) of 75.6%, and an F ratio ($F = \frac{2PR}{P+R}$) of 74.4%.

For all-spans-training/all-spans-validation on our AMT dataset, the best model had a regularization parameter $C = 0.5$ using L2-regularization with L2-loss SVC (dual). With these parameters, TP = 50.39%, FP= 11.39%, FN = 15.46%, TN = 22.75%, resulting in an overall accuracy of 73.14%, precision of 81.56%, recall of 76.52%, and an F ratio of 79.0%.

[2]Please visit http://www-nlp.stanford.edu/∼jeaneis/dokuwiki-2012-10-13/doku.php?id=side-by-side for comparison.

Train	Valid	Label	MNB-uni	SVM-uni	MNB-bi	SVM-bi
root	root	RT	75.2	72.8	75.9	74.5
root	root	AMT	74.6	72.5	73.8	74.8
all spans	root	AMT	77.1	76.3	n/a	n/a
all spans	all spans	AMT	75.2	72.3	n/a	n/a

Table 1: Accuracy of MNB and SVM algorithms. Label Source: AMT - *Amazon Mechanical Turk*, RT - *Rotten Tomatoes*. "Root" indicates sentiments of entire excerpts only, while "all spans" indicates sentiments of all sub-spans of the excerpts. For the bigram models, only the sentence-level data are available.

	MNB-uni	**SVM-uni**
1	75.9	72.6
2	75.6	72.7
3	76.4	73.5
4	76.6	73.5
5	76.7	73.8
6	77.0	74.0
7	76.0	72.5
8	76.4	73.3
9	77.2	74.2
10	76.1	72.4
Average	76.4	73.3
Wang	77.9	76.2

Table 2: Results of MNB and SVM on 10-fold cross-validation, with comparison to (Wang & Manning, 2012).

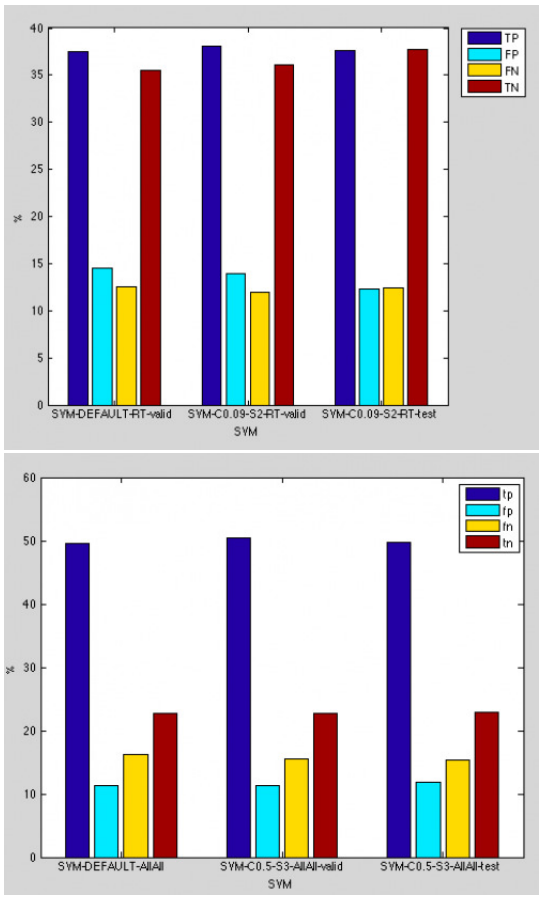

Figure 4: Top: Plotting the True Positives (TP), False Positives (FP), False Negatives (FN) and True Negatives (TN) for SVMs trained and validated on root/root RT data. Bottom: SVMs trained and validated on all-span/all-span AMT data.

Results from Deep Learning Our results showed that this particular method of deep learning achieved approximately 60% accuracy on the test set using the binning metric with or without using different U and $b^{(2)}$ for different values of n. These results show that our neural network does not perform as well as the SVM and MNB classifiers that use unigrams and bigrams to determine sentiment. The figure below shows the changes in accuracy when we were tuning our parameters - H, C (the regularization constant), and the number of passes (called numPass) through our training set.

From Table 3, we can see that the accuracies fluctuated within a small window around 60% with some of the more extreme values of the parameters causing larger decreases in accuracy. Looking at our gradients, we can see that our choice of H would affect our regularization parameter C because we would have different values for vectors of different lengths and, as a result, different gradient values. Additionally, the number of passes through the data affects the updates to our parameters; it is interesting to see that, with our network, a larger number of passes did not improve the performance of our network. From trying out various combinations of H, C, and numPass, we found that our best validation accuracy was 0.6167 with $H = 100$, $C = 1$, and $numPass = 10$, and applying this model on our test set yielded an accuracy of 0.6130.

As expected, our 3-layer neural network performs well on unigrams and bigrams but starts to make more mistakes as n becomes large. The reason for this is a combination of having fewer training examples as well as having an increasing structural complexity for larger n. Figure 5 shows the accuracy versus the size of the n-gram as well as the number of testing n-grams.

These accuracies are from binning the values from the output layer into the (# of ratings) bins on our 0 to 1 scale. When we used intervals proportional to the fraction of each rating seen during training to better represent the rating scale, we achieved a significantly higher accuracy of 95% on our validation and test sets. This improvement is more of a consequence of the nodes in our validation and test sets having mostly 3-star (neutral) ratings and relatively very few 1-star, 2-star, 4-star, or 5-star labels, so our increase in accuracy is due to the fact that our a priori probabilities provide a tremendous amount of help in correctly binning the predicted scores. Instead of binning the output from our logistic regression classifier and using harsh assumptions about the splits in our output probabilities, we could alternatively use a softmax classifier as a more accurate way of determining the appropriate labels in our output layer.

H	C	numPass	Accuracy
50	1.0	10	0.5165
100	1.0	10	**0.6167**
125	1.0	10	0.5916
150	1.0	10	0.6144
175	1.0	10	0.5821
200	1.0	10	0.6084
100	0	10	0.5823
100	0.1	10	0.5931
100	0.5	10	0.5428
100	0.8	10	0.5870
100	**1.0**	10	**0.6167**
100	1.5	10	0.5740
100	2.0	10	0.5910
100	5.0	10	0.5984
100	1.0	5	0.5692
100	1.0	8	0.6041
100	1.0	**10**	**0.6167**
100	1.0	12	0.5928
100	1.0	15	0.5976

Table 3: Parameter Tuning: Accuracy on the validation set for various combinations of H, C, and numPass values. The best model is $H = 100$, $C = 1.0$, and numPass=10 with an accuracy of 0.6167

Rating	# Correct	# Predicted	# Actual
1	25	408	2012
2	1512	8383	9414
3	47,768	64,347	57,431
4	2083	10,442	11,188
5	39	316	3851

Table 4: Number of correctly labeled test nodes along with the number of predicted labels and actual labels grouped by rating of 1 star to 5 stars.

Error Analysis for Deep Learning Using the best model identified during parameter tuning on the test set, we compared our predicted labels to the actual labels to find the precision and recall by rating. The results are shown in Table 4.

We can see that, because the data set itself our training sets and test sets have a neutral sentiment rating, our model has a higher recall for a rating of 3 stars but not for any other rating. A qualitative explanation for these numbers is that our model has learned that, in order to maximize the likelihood, it should err on the side of labeling something as neutral unless there is an extremely positive or negative predicted label. As a result, our precision for these n-grams is higher for the non-neutral n-grams. Looking at the # Predicted and # Actual columns in Table 4, we can see that our model is extremely unwilling to label n-grams with 1-star or 5-star ratings. The graph shown in Figure 6 suggests that our precision and recall

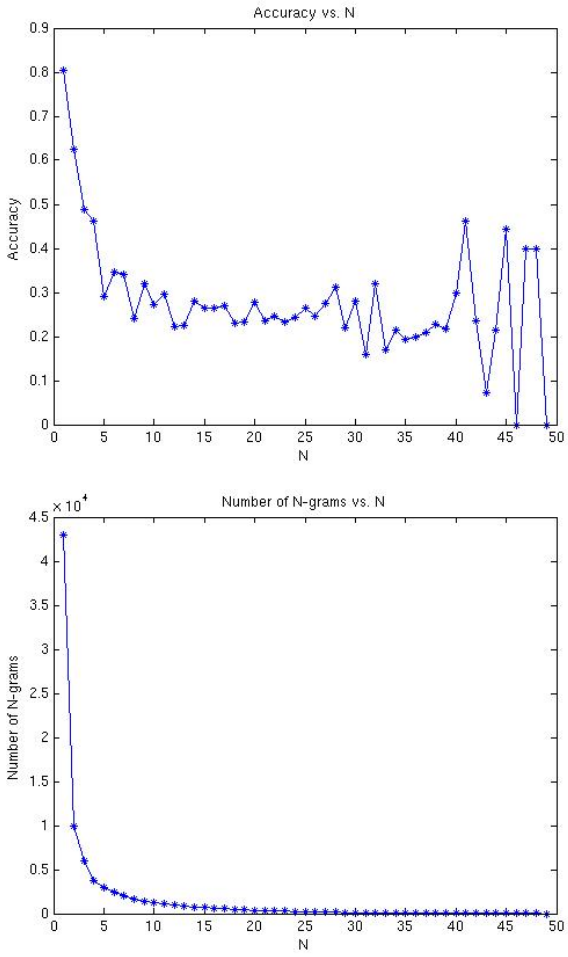

Figure 5: Top: Accuracy vs. N. Bottom: Number of Test N-grams vs. N

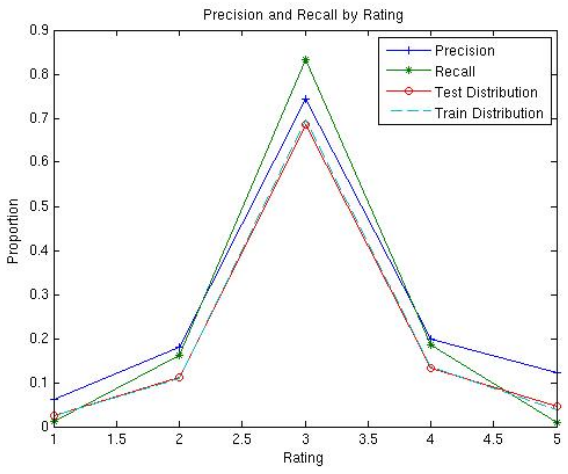

Figure 6: Precision and recall percentages by rating of 1 star to 5 stars with the distribution of actual labels seen in the test set.

curves follow the distribution of labels for our training and test set; this potentially implies that, if our training examples were more evenly distributed across the ratings, we may have a better model for distinguishing between them.

From another perspective, when we looked at the precision and recall compared to the length of the n-gram, we noticed that the precision for n-grams with neutral ratings showed a steady decrease from 0.9 to 0.3 as n increased, whereas other precision values generally fluctuated about 0.2 with a few spikes. As discussed earlier, the precision for ratings of 1 or 5 were mostly zero because our model rarely assigns that label to any n-gram. The recall values for various n seemed rather arbitrary because it depends on the behavior of the test labels, but the one thing that remained consistent was that the recall values for ratings 1 and 5 were also mostly zeros.

Our model performed best on unigrams and bigrams, but occasionally it would make errors on words or phrases that would require more context to confirm the sentiment. Some examples of these mistakes were the following:

1. *Incorrectly Labeled Unigrams*: new, splash, greater, good, place, rare, movie, fun, too-tepid, Effective

2. *Incorrectly Labeled Bigrams*: neurotic mindset, something rare, Arnold Schwarznegger, so honest, issue movie, Effective but, a splash

And on the other end, we correctly labeled some longer n-grams that had non-neutral ratings:

1. "to be the 21st Century 's new " Conan " "

2. " doing what it does with a dedicated and good-hearted professionalism "

3. " held the Enterprise crew together through previous adventures and perils "

4. " A gentle , compassionate drama about grief and healing . "

5. " The film has a few cute ideas and several modest chuckles but it is n't exactly kiddie-friendly ... Alas, Santa is more ho-hum than ho-ho-ho and the Snowman (who never gets to play that flute) has all the charm of a meltdown "

6. " A clutchy and indulgent and pretentious travelogue and diatribe against ... well just stuff . Watching Scarlet Diva, one is poised for titillation, raw insight or both . Instead, we just get messy anger, a movie as personal therapy . "

Although the performance for longer n-grams was not good, we can see that some sentiment is captured. For instance, the fifth example listed above contains many seemingly positive words such as *cute*, *chuckles*, or *charm*, but our model manages to combine it with the negative phrases like *meltdown* or *isn't exactly* to give it a negative predicted label.

Conclusion and Further Work

We can conclude that, with regards to achieving a performance better than that of our MNB and SVM models, our current 3-layer neural network is insufficient. Our results showed that this network performed well on unigrams and bigrams but suffered for most larger n-grams. From our error analysis, we can see that we are not taking the constituents of the larger n-grams into account despite having the parse structure available in our data set. In unigrams and bigrams, this effect is not as noticeable because the constituents are the word vectors, but we can attribute the drop in accuracy going from unigrams to bigrams to the fact that we do not consider how certain words affect their neighboring words (e.g. *neurotic* combined with *mindset*). Therefore, an improved neural network could include representations for how words act on their neighbors as well as which types of edges were used to construct the parse of the n-gram.

Another way of viewing this problem is from a language perspective — the meaning of a sentence is built from its components, and its sentiment is an aggregation of the sentiments of those components. By identifying each n-gram from a parse tree and treating them as independent n-grams, we are disregarding the information that we learned from the networks for the smaller n-grams when we create the model for the larger ones. Our MNB and SVM models utilizes this characteristic of language and sentence construction to obtain a smaller classification error. There are many approaches to address this issue; one idea is like the one we mentioned before with encoding representations for edges as well as words while another idea is to use a recursive neural network and have each node in the parse trees output to a hidden layer from the hidden layer output from its two children which we then use our classifier to compute the predicted class label.

Deep learning has a lot of potential for sentiment analysis, but our results here proved that a basic neural network with only word vector representations as features will perform worse than other classification techniques that build upon just bigram and unigram data. Current research in machine learning is focusing on new extensions of and modifications to the basic structure and model to cover a wider range of learning tasks, and hopefully these developments will allow us to find the right way to encode a superset of the behaviors and details that other techniques cannot fully or properly capture to achieve the best performance and gain insight into the learning goals.

Acknowledgments

We would like to thank Richard Socher for providing guidance and for giving us a good starting reference for optimizing deep learning code (Huang, Socher, Manning, & Ng, 2012). We would also like to thank Chris Manning, and Sida Wang for helpful suggestions and advice. The sentiment collection of the dataset was jointly funded by Chris Potts and Andrew Ng.

References

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., & Lin, C.-J. (2008, June). Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, *9*, 1871–1874.

Golder, S. A., & Macy, M. W. (2011). Diurnal and seasonal mood vary with work, sleep, and daylength across diverse cultures. *Science*, *333*(6051), 1878-1881.

Huang, E. H., Socher, R., Manning, C. D., & Ng, A. Y. (2012). Improving Word Representations via Global Context and Multiple Word Prototypes. In *Annual meeting of the association for computational linguistics (acl).*

Klein, D., & Manning, C. (2003). Accurate unlexicalized parsing. In *Proceedings of the 41st annual meeting on association for computational linguistics-volume 1* (pp. 423–430).

Pang, B., & Lee, L. (2005). Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Annual meeting-association for computational linguistics* (Vol. 43, p. 115).

Wang, S., & Manning, C. (2012). Baselines and Bigrams: Simple, Good Sentiment and Topic Classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics.*