Sentiment Analysis on Movie Reviews

COSC-572/LING-572 ENLP Course Project[1]

Mitchell Abrams
Department of Linguistics

Shaobo Wang, Zhuoran Wu
Department of Computer Science

Georgetown University
{mja284, sw1001, zw118}@georgetown.edu

Abstract

In this study, we compare the effectiveness of various machine learning models for a sentiment analysis task. The data is from the Rotten Tomatoes movie review corpus and we aim to classify each review to three labels: ‘negative’, ‘neutral’, ‘positive’. We apply a series of natural language processing methods to extract features from text data. We also demonstrate that for this task, Support Vector Machines perform the best and recurrent neural networks LSTM is trailing close behind. We then show that LSTM performs better with label five class labels from the Kaggle competition standards. We conclude by explaining why these models are unable to reach relatively high accuracy for this dataset and the direction for future work.

1 Introduction

For this project, we apply a series of machine learning models to analyze the sentiment for each movie review. The corpus is from the rottentomatoes.com website and originally collected and published by Pang and Lee (2005). Sentiment analysis is a topic that has a wide range of literature and has gained a lot of traction in field of NLP. The internet has afforded researchers the opportunity to collect large amounts of language data to train machine learning models for sentiment analysis among other classification tasks. Sentiment analysis falls under the larger scope of opinion mining and seeks to extract the opinion or sentiments for a particular entity or target. Although this is a prevalent topic in NLP, sentiment analysis is a difficult task as language is complex and opinions can be expressed in different ways. Common challenges include handling negation, humor/irony, ambiguity, and capturing the target of an opinion.

Attempts to handle these challenges have been addresses in the sentiment analysis literature. With the various approaches in mind, we test an array of machine learning methods and apply a unique set of feature engineering to obtain a high accuracy and compare our results to state-of-the-art methods and results.

[1] Project Github Repo: https://github.com/sw1001/ENLP-Project

In this paper we will introduce literature that guided our study and its relevant to our specific task. Then we will proceed to cover the data we trained our models on. Next we will describe our models and feature engineering and conclude by discussing our experiment and results.

2 Related Work

The first paper we have reviewed is "*Thumbs up? Sentiment Classification using Machine Learning Techniques*" by Bo Pang et al. (2002). In this paper, the authors presented several prior-knowledge-free supervised machine learning methods to handle the sentiment analysis problem. The authors presented a baseline experiment. It asked two human reviewers to manually classify the reviews by a proposed positive and negative word list. The accuracy can be up to 69%. The study indicates that knowledge-based methods, with manual construction, can't always guarantee a better results.

Then the authors presented the machine learning based method. They selected Naive Bayes (NB), Max Entropy (ME), and Support Vector Machine (SVM) as the models because the philosophies behind these three are quite different, while each has been shown to be effective in previous text categorization studies. So their results are worth comparing. For feature engineering, the authors applied a standard bag-of-features framework. They calculated unigrams, bigrams, POS, and the position of a word in the text. They also converted the frequency of a feature into presence by binarizing the feature to 1 if it existed in the document, 0 otherwise.

	Features	# of Features	Frequency or Presence	NB	ME	SVM
(1)	Unigrams	16165	freq.	78.7	N/A	72.8
(2)	Unigrams	16165	pres.	81.0	80.4	82.9
(3)	Unigrams+Bigrams	32330	pres.	80.6	80.8	82.7
(4)	Bigrams	16165	pres.	77.3	77.4	77.1
(5)	unigrams+POS	16695	pres.	81.5	80.4	81.9
(6)	Adjectives	2633	pres.	77.0	77.7	75.1
(7)	Top 2633 unigrams	2633	pres.	80.3	81.0	81.4
(8)	Unigrams+Position	22430	pres.	81.0	80.1	81.6

Tab 1. Average three-fold cross-validation accuracies, in percent, by Bo Pang et al.

As shown in the result table, NB tends to do the worst and SVMs tend to do the best, although the differences weren't very large. Unigram presence information turned out to be the most effective. The accuracy can reach as high as 82.9%, which is much better than human-based reviewers. Guided by this study, we implement a Naive Bayes model for our project first as a baseline. Then we derive similar features as mentioned in the paper to see how its performance on our data set. We also test if stratifying the data set can help on improving the results.

In Bo Pang and Lillian Lee's paper "*A sentimental education: Sentiment analysis using subjectivity*", we utilize an architecture which is shown below:

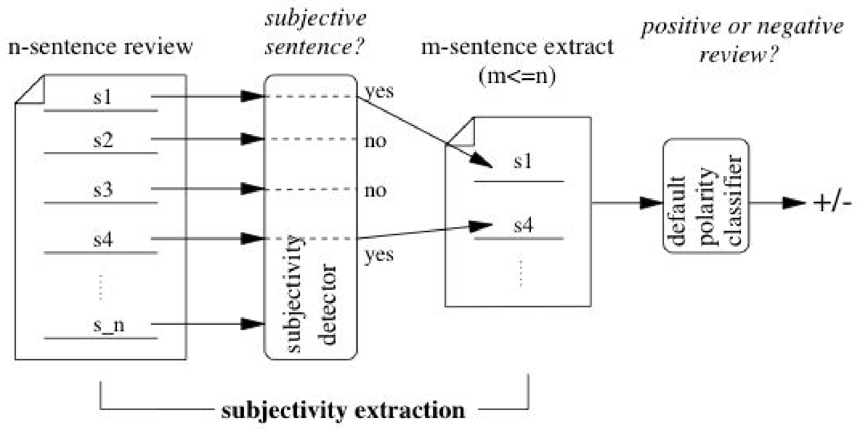
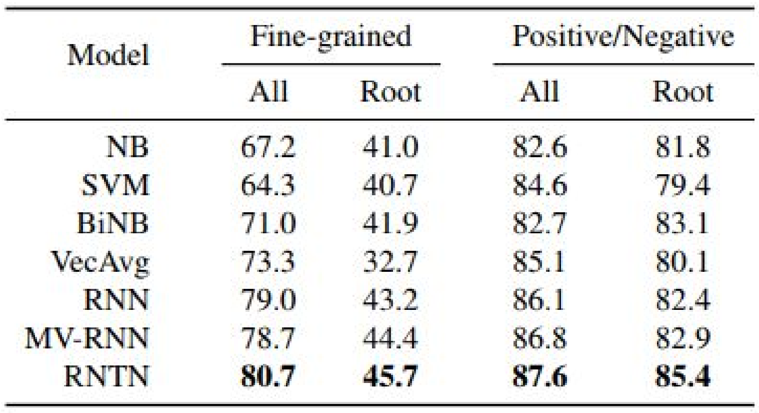


Fig 1. Polarity classification via subjectivity detection presented by Bo Pang and Lillian Lee

We use text processing tools to analyze the subjectivity and polarity for a sentence to create a program based on this architecture to see how these tools work on our data set.

In the paper "*Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales*" by Pang and Lee (2005), the authors address the rating-inference problem. Rather than simply deciding whether a review is "positive" or "negative" as in previous sentiment analysis work, one must determine an author's evaluation with respect to a multi-point scale (e.g., one to five "stars"). The methods introduced in the paper are designed for multi-category classification problems, which are similar to the original Kaggle competition. Due to the constraint of time limit, we decided to simplify the task. We only borrow the idea of using SVMs as a machine learning method for our project.

The paper "*Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank*" by Richard Socher et al. (2013) from Stanford University presents an innovative method of using sentiment treebank with deep learning models. 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 same movie review corpus we are using. The authors propose a new model called the Recursive Neural Tensor Network (RNTN) which achieves the highest accuracy in their experiments. Also, by applying treebank, RNTN also achieves the best accuracy on binary sentiment considering negation. In fact, the combination of the new sentiment treebank and the RNTN pushes the state of the art on short phrases up to 85.4%.



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**

Tab 2. Accuracy for fine grained (5-class) and binary predictions at the sentence level (root) and for all nodes.

Building a sentiment tree is complicated and requires other resources as well. We simply use two libraries: pytreebank and Jupyter to illustrate a sentiment tree.

The series of lectures, "*Sentiment Analysis and Opinion Mining",* Morgan & Claypool Publishers, by Liu, Bing (2012) provides us with a comprehensive overview of sentiment analysis and the important topics and developments in the field. This survey is helpful for understanding general trends in sentiment analysis research and possible challenges. We have made use of some important points from this survey. A major one is an in-depth definition of an opinion so we can understanding its components. Opinions are classified as regular and comparative, and within regular opinions, direct and indirect. This helps us to focus on what is important for our task and the types of structures we would expect to see; opinions contain a topic, sentiment, an opinion holder and time. This shows that the definition of an opinion is complex and that many aspects that it's difficult to capture all of these aspects with our present classification methods. One of the studies in this survey is Snyder et al. (2007) which attempts to capture aspects of an opinion where other studies such as Bo Pang et al. (2002) assume there is one opinion in a text.

For our purposes, we assume there is one opinion for each review corresponding to the scale of sentiment. So we will follow traditional approaches by combine lexical features such as unigrams, bigrams, trigrams, and TF-IDF in our classification task. Varying N-grams for instance can capture negation like "not good" (bigram) or "not very good" (trigram).

Within the sentiment lexicon generation section, Liu (2012) covers sentiment lexicon generation and sentiment subjectivity/polarity, features we use in our baseline with Textblob (a Python library for processing textual data). This helps us to understand how it works and why it isn't the best feature by itself considering sentiment orientation is content and context dependent (Liu 2012, pg:43). Lastly, this paper outlines some common models used such as SVM and Naive bayes, which we implement in our project.

3 Exploratory Data Analysis

Before we do the feature engineering, we need to take a deep look to the data. Our data provide movie reviews from a dataset that comes from the Rotten Tomatoes movie review corpus, with parsed phrases labeled by human annotators. The original data is annotated for five labels: 0 - negative, 1 - somewhat negative, 2 - neutral, 3 - somewhat positive, 4 - positive. But for our project we collapse this to three labels: 0 - negative, 1 - neutral and 2 - positive. The training set and test set are files in .tsv format. The training data is organized by columns labeled 'PhraseId', 'SentenceId', 'Phrase', and 'Sentiment'. Below are the first few rows of the original training set:

PhraseId	SentenceId	Phrase	Sentiment
1	1	A series of escapades demonstrating the adage that what is good for the goose is also good for the gander , some of which occasionally amuses but none of which amounts to much of a story .	1
2	1	A series of escapades demonstrating the adage that what is good for the goose	2
3	1	A series	2

Tab 3. Movie Reviews Dataset Shortcut.

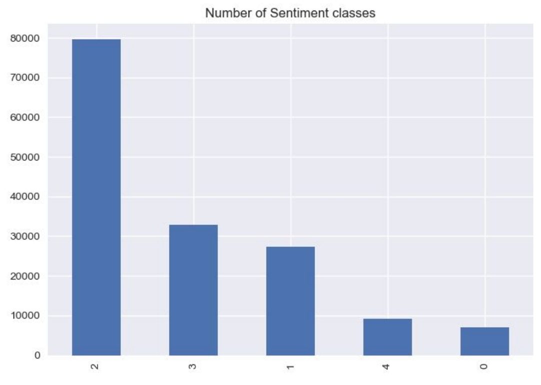
There are in total 156060 sentences and phrases in the training set. Specifically, there are 8544 complete sentences.

Then we extract the most common positive and negative words with N-gram Model. The result is same as our experience. People would like to use similar word such as "Best", "Excellent", "Bad", "Worst", or the similar word phrases such as "A wonderful movie", "Complete failure".

n	Most positive Common Words	Most Negative Common Words
1	Good, Best, Love, Brilliant, Excellent	Bad, Dreary, Worst, Hard
2	A masterpiece, Fantastic Movie, Wonderful performance	Worst Movie, Worthless film, Complete Failure
3	A wonderful movie, A small gem	A lousy movie, a complete failure

Tab 4. Most Common Words in both classes with N-gram Model

Then we draw the distributions of the dataset and next we use WordCloud to build an image for the positive class:



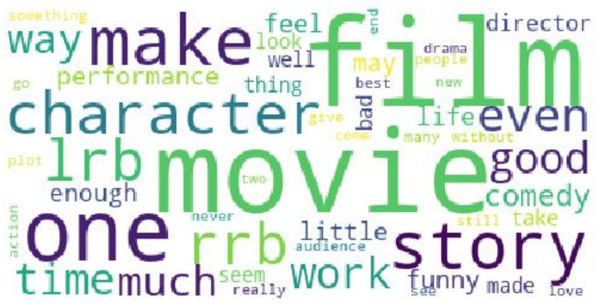


Fig 2. (Left) Numbers of sentences and phrases in 5 classes.
Fig 3. (Right) Most Frequent Words in Positive Class.

From the the distribution, we could see that it is an imbalanced database where 2 ("Neutral") is the most of the phrases' sentiment.

4 Feature Engineering

4.1 Data Preprocessing

To process our corpus we implemented the following cleaning measures to ensure that our models were working with relevant text. We collected the raw text of the movie reviews and converted each chunk unto a list of cleaned words. The process begins with using regular expressions to substitute non alphabet letters with a space, reducing noise in the corpus like punctuation which is usually not relevant for sentiment. We also lower cased our text to make it more consistent. From NLTK (Bird, et al., 2009) we used WordNetLemmatizer to reduce morphological variation and removed stop words for certain tasks. By collapsing morphological forms for a feature vector, we reduce sparsity and allow one feature to represent many variations

4.2 Features

4.2.1 N-grams

An n-gram model is a type of probabilistic language model for predicting the next item in such a sequence in the form of a $(n - 1)$–order Markov model.[2] n-gram models are now widely used in probability, communication theory, computational linguistics (for instance, statistical natural language processing), computational biology (for instance, biological sequence

[2] Introduction to N-grams: https://web.stanford.edu/class/cs124/lec/languagemodeling2017.pdf

analysis), and data compression. Two benefits of n-gram models (and algorithms that use them) are simplicity and scalability – with larger n, a model can store more context with a well-understood space–time tradeoff, enabling small experiments to scale up efficiently.

4.2.2 BOW features

Bag of Words (BoW)[3] is a model used in natural language processing. One aim of BoW is to categorize documents. The idea is to analyse and classify different “bags of words” (corpus). And by matching the different categories, we identify which “bag” a certain block of text (test data) comes from.

4.2.3 WordNet

WordNet is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. WordNet superficially resembles a thesaurus, in that it groups words together based on their meanings. However, there are some important distinctions. First, WordNet interlinks not just word forms—strings of letters—but specific senses of words. As a result, words that are found in close proximity to one another in the network are semantically disambiguated. Second, WordNet labels the semantic relations among words, whereas the groupings of words in a thesaurus does not follow any explicit pattern other than meaning similarity.[4]

4.2.4 General Inquirer lexicon[5]

The Harvard General Inquirer is a lexicon attaching syntactic, semantic, and pragmatic information to part-of-speech tagged words (Stone et al., 1966). The spreadsheet format[6] is the easiest one to work with for most computational applications. Table 3 provides a glimpse of the richness and complexity of this resource.

	Entry	Positive	Negativ	Hostile	...184 classes...	Othtags	Defined
1	A		Negativ			DET ART	...
2	ABANDON		Negativ			SUPV	
3	ABANDONMENT		Negativ			Noun	

Tab 5. Harvard General Inquirer Data Shortcut.

[3] Bag of Words (BoW) - Natural Language Processing: https://ongspxm.github.io/blog/2014/12/bag-of-words-natural-language-processing/
[4] WordNet https://wordnet.princeton.edu/
[5] General Inquirer lexicon http://www.wjh.harvard.edu/~inquirer/
[6] Spreadsheet Format http://sentiment.christopherpotts.net/lexicons.html#inquirer

4.2.5 Sentiment Treebank

Stanford Sentiment Treebank[7]—an extension of MR but with train/dev/test splits provided and fine-grained labels (very positive, positive, neutral, negative, very negative), re-labeled by Socher et al. (2013).

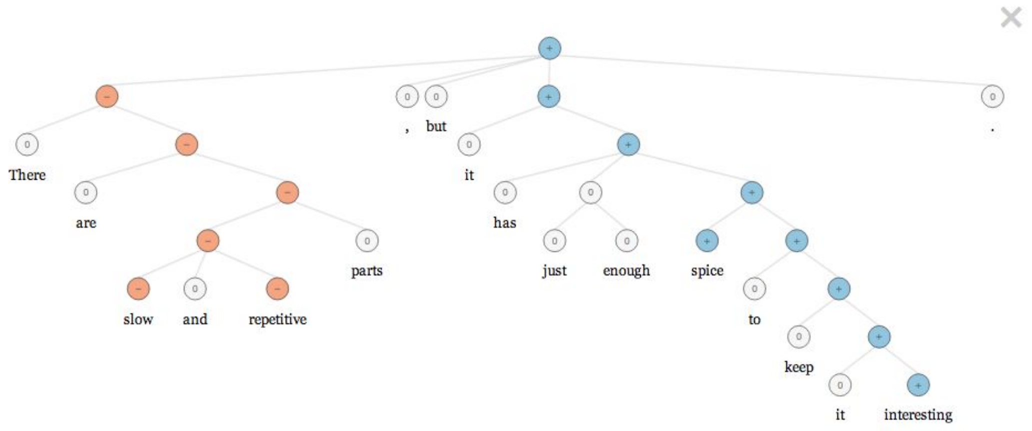


Fig. 4. A Example of Sentiment Treebank.

4.2.7 Word Embedding

Word embeddings are distributed representations of words, represented as real-valued, dense, and low-dimensional vectors. Each dimension potentially describes syntactic or semantic properties of the word. Here we briefly describe the three types of pre-trained embeddings that we use in our work.[8]

Mikolov et al. (2013a)[9] propose two log-linear models for computing word embeddings from large corpora efficiently: (*i*) a bag-of-words model CBOW that predicts the current word based on the context words, and (*ii*) a skip-gram model that predicts surrounding words given the current word. They released their pre-trained 300-dimensional word embeddings (vocabulary size 3M) trained by the skip-gram model on part of Google news dataset containing about 100 billion words[10].

[7] Summary of "Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank" https://gist.github.com/shagunsodhani/6ca136088f58d24f7b08056ec8b97595
[8] Pengfei Liu, Shafiq Joty, Helen Meng, Fine-grained Opinion Mining with Recurrent Neural Networks and Word Embeddings, Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1433–1443.
[9] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. In Proceedings of Workshop at ICLR, 2013.
[10] Word2Vec, https://code.google.com/archive/p/word2vec/

5 Experiments

5.1 Experiment Set-up

There are **8500+** whole sentences in the dataset. However, the dataset provides **140,000+** phrases that contains whole sentences with their sentiment. The sentiment range varies from **0-4** corresponding to negative, somewhat negative, neutral, somewhat positive and positive.

In order to simply our task, we map all “Somewhat negative” to “negative” and “Somewhat positive” to “positive” and sentiment range varies from **0-2** corresponding to **negative, neutral and positive**. Then we extract all the **whole sentence** rows from original dataset, with the mapped data. Then we apply the **data cleaning process** to the dataset which includes removing stop words, word lemmatization, etc.

Here is the explanations for the different dataset:

- 8530 Sentences
  Only contains all whole sentences and mapped sentiment without data clean.

- Cleaned 8530 Sentences
  Only contains all whole sentences and mapped sentiment with data clean

- Whole Dataset
  Contains whole dataset and mapped sentiment without data clean.

- Cleaned Whole Dataset
  Contains whole dataset and mapped sentiment with data clean.

About Evaluation Methods, no matter which dataset we use after feature engineering, we always split the data into train and test data. Currently, we set 0.2 as **test data size**. Furthermore, we did the **shuffle** and **resampling** process.

Considering this to be a multiclass classification problem, we use **accuracy**, **precision**, **F1**, **recall** metrics to evaluate our model. Inside our deep learning method, we use **mlogloss** as our evaluation method.

For each experiment, we will output all those evaluation results along with a **confusion matrix**. However, in the following part, we only provide the **accuracy** as our output. The accuracy scores function is sklearn.metrics.accuracy score. Also, in further experiments, we will also use **Cross Validation** as our result.

Model	Test Acc	F1	Recall	5-Fold CV
Baseline Majority	42.09%	None	None	None
Baseline Textblob	57.%	0.58	0.54	None
XGBoost	54.10%	0.54	0.49	None
Naive Bayes	68.33%	0.69	0.69	68.2%
SVM	**73.86%**	**0.74**	0.74	71.2%
Random Forest	72.49%	0.72	0.73	70.6%
Ensemble LSTM	72.73%	0.73	0.73	None
LSTM	70.67%	0.73	**0.75**	68.32%

Tab 6. Experiment Results on 3 classes.

Model	Kaggle Test Acc	10-Fold CV
Majority Baseline	0.411	None
SVM	0.622	None
LSTM	**0.643**	60.86%
Ensemble Random Forest	**0.658**	61.32%
Kaggle Current Best[11]	0.765	None

Tab 7. Experiment Results on 5 classes on Kaggle.

5.2 Result Discussion

Compared with majority baseline, our results with machine learning method are quite good. Among those technologies, SVM and LSTM perform almost better than other methods. Detailed experiment results are listed in Appendix 1. If we used the same features such as N-grams, we find that Naive Bayes performs almost worse than the SVM. This conclusion is comparable to many papers [12]discussed. Adding more features leads to overfitting our language model. Furthermore, Unigram presence information turned out to be the most effective; in fact, none of the alternative features we employed provided consistently better performance once unigram presence was incorporated (Bo Pang et al., 2002).

[11] Current Best Score: 0.76526: https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews/leaderboard

[12] Bo Pang, Lillian Lee, Thumbs up? Sentiment Classification using Machine Learning Techniques, Proceeding of the Conference on Empirical Methods in Natural Language Processing, Philadelphia, July 2002, pp. 79-86

On the other hand, we found that Random Forest will perform slightly better if we use General Inquirer (GI) Lexicon and WordNet features to construct a sentence vector. It shows that features based on word semantics could be better than the features based on word frequency.

Furthermore, we found that RNN-LSTM will perform better than traditional machine learning methods even if we use the most basic embedding method with Text-to-Sequence.

Based on those two findings, we would like to use Semantic Word Embedding on Deep Learning Methods. Word Embedding is a method that map word sequence to number vectors. Now, there are papers that discuss how to implement learning semantic word embeddings based on corpus and knowledge base.

Last but not least, we also apply ensemble averaging method to enhance our performance. We can take advantage of the fast convergence of the LSTM method by training multiple models and ensembling the predictions. According to our results, we could see that ensemble-averaging leads to an about 1.5% improvement in accuracy.

After we compare the accuracy result for different models, we start to focus on why the models make wrong predictions. To look at this clearly, we print out some instances of False Negatives. We found that common phenomenon in the documents is a "thwarted expectations" narrative, where the author sets up a deliberate contrast to an earlier discussion. For example, "***However stale the material , Lawrence 's delivery remains perfect ; his great gift is that he can actually trick you into thinking some of this worn-out, pandering palaver is actually funny .***" (Negative) In these examples, a human would easily detect the true sentiment of the review, but our features classifiers would presumably find these instances difficult, since there are many words indicative of the opposite sentiment to that of the entire review.

Also, we found that the dataset itself is not good enough in classify the sentiment of reviews. For example "***The movie should jolt you out of your seat a couple of times, give you a few laughs , and leave you feeling like it was worth your seven bucks , even though it does turn out to be a bit of a cheat in the end.***" (Neutral) It tends outs to be negative or at least a somewhat negative review. However, the label itself it will show the bias of classification. We suspect that the Kaggle competition didn't provide a good problem description. We think it would be better if the movie reviews can be categorized into a scale from one star to five stars, where a neutral review is not "sentimentally neutral" any more. Instead, it can be considered as a three-star review which expresses an intermediate attitude/opinion to a movie. On the other hand, splitted phrases can be considered in a 0 to 4 scale as before. Because usually they don't express any semantics with reasonable logic. It would make more sense if we only label them by their subjectivity/polarity.

6 Conclusion

Our experiment demonstrates that an SVM model works best for this specific task; predicting sentiment of three labels (‘negative’, ‘neutral’, ‘positive’) on the Rotten Tomatoes data set. Our Random Forest model and Ensemble LSTM also have similar accuracies to the SVM model. This demonstrates that the SVM method can work well with this amount of training data and a bag-of-words approach. With more training data we expect our neural networks to perform better.

This study has proved to be challenging as various methods we employed were unable to reach state-of-the-art accuracy and the winning Kaggle accuracy score. While we experimented with various linguistic features, many of them failed to yield better results. It proved difficult to capture certain sentences that have “thwarted expectations” or use humor to express opinion. This is because many of our approaches depended on individual words and shorter contexts. We found that using higher order linguistic features such as synset vectors and lexicon based approaches boosted Random Forest performance, and unigram features worked well with more traditional machine learning techniques.

Future directions using this same data set would be to apply aspect engineering; extract entities from a text and map opinions to their targets. Another possible approach would be to leverage discourse features as applied by Benamara et al. (2017) and parse larger text into discourse relations. The intuition behind this approach is that rhetorical relations (CONCESSION, CONTRAST, CORRECTION, SUPPORT, etc.) whether explicit or implicit, can contribute to understanding the sentiment of a text with a larger scope.

Lastly, something we learned from this study is that while certain methods tend to do better than others in sentiment analysis, the task is largely dependent on the corpus used to train the models. Language in one domain will be highly different than another domain and labels might not be correct. More data will also train models to perform better by exposing it to more examples and variation. This is why it might be important to have expert annotated corpora and collect more language data that captures sentiment in various ways.

7 Reference

Benamara, Farah, Maite Taboada, and Yannick Mathieu. *"Evaluative language beyond bags of words: Linguistic insights and computational applications." Computational Linguistics* 43.1 (2017): 201-264.

Bird, Steven, Edward Loper and Ewan Klein (2009), *Natural Language Processing with Python*. O'Reilly Media Inc.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan, Thumbs up? Sentiment Classification using Machine Learning Techniques, Proceedings of EMNLP 2002.

Bo Pang and Lillian Lee, A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts, Proceedings of ACL 2004.
McCallum, Andrew, and Kamal Nigam. *"A comparison of event models for naive bayes text classification." AAAI-98 workshop on learning for text categorization*. Vol. 752. No. 1. 1998.

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

Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." *Journal of machine learning research* 12.Oct (2011): 2825-2830.

Snyder, Benjamin, and Regina Barzilay. *"Multiple aspect ranking using the good grief algorithm." Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*. 2007.

Socher, Richard, et al. "Recursive deep models for semantic compositionality over a sentiment treebank." *Proceedings of the 2013 conference on empirical methods in natural language processing*. 2013.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. In Proceedings of Workshop at ICLR, 2013.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. In Proceedings of NIPS, 2013.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic Regularities in Continuous Space Word Representations. In Proceedings of NAACL HLT, 2013.

8 Appendix 1 - Methodology

In this section we will briefly describe the supervised machine learning techniques that we employed in our study. Each of the models are distinct and deserve an explanation to illustrate how they might be useful this task. In following the classification experiment in Bo Pang et al. (2002), we used used a Naive Bayes model and an SVM to classify movie reviews. We also compare decision tree algorithms and neural networks, namely Random Forest, XGBoost, and RNN-LSTM. All of our models are used off the shelf from the Scikit Learn library (Pedregosa et al 2011).

8.1 Naive Bayes

The Naive Bayes model is one of the machine learning models we implemented in our study as it has been used for sentiment analysis (Pang and Lee, 2002). In Pang and Lee's study, the Naive Bayes performed the worst model on their IMDB movie review set. Nevertheless, we wanted to incorporate this model to assess how it would perform on the rotten tomatoes data set and see if we could tune the model for better performance.A brief explanation of the model and its strengths will help to illustrate how it performs and the decisions it makes.

The Naive Bayes algorithm is a generative model used to predict the likelihood that an event will occur given data its given. This is in contrast to a discriminative model that fits with gradient descent. In our case, the model is predicting the sentiment of a review given the training corpus of labeled data.

There are three types of Naive Bayes models: Multinomial, Bernoulli, Gaussian. For our study we focus on the Multinomial model at it is more appropriate when dealing with discrete values like n-grams. Bernoulli and Gaussian versions of the Naive Bayes are used when working with continuous data like a time series. This decision was informed by McCallum et al. (1998) that showed how the multinomial model perform better than a Bernoulli model in a text classification task with similar features.

Another important aspect of this model is that predictors are independent of each other- this is a “bag of words” approach. Although we are capturing order of words to an extent with bigrams and trigrams, order is largely ignored. The Naive Bayes assumption, which this model relies on, assumes the words are independently conditioned on the class.This assumption contradicts how language actually works but it has proven to be effective for classification tasks.

Features used to create our feature vectors include unigram, bigram, and trigrams. Unigrams can capture salient sentiment words like “good” or “bad” while bigrams and trigrams can naively capture simple negation. The model chooses the class that makes the review most probable. The model can be tunes by testing combination of features and adjusting smoothing value. Smoothing helps to avoid over fitting our data.

Some weaknesses we expect to see from this model is it's failure to take longer distance affects into account, namely, long distance negation, discourse features that add to the nuance of the sentiment, opinion targets, and humor. As a result, the accuracy will be higher than the baseline but not state-of-the-art.

8.2 SVM

Another model we employed was a Support Vector Machine. An SVM is a discriminative model that is not used to predict the class directly. The model draws a hyperplane in vector space to separate points representing the classes.The line drawn to separate the data in training is adjusted until optimization is achieved and it maximizes the margin between points from each class. Points can be allowed to cross boundaries in order to avoid overfitting the data. In this study we use a linear kernel to separate the data but data may not always be linearly separable. A kernel trick allows the model to separate data in a different way by mapping it to higher dimensions.

This was the best performing model in (Pang and Lee, 2002) and we expect higher accuracy when compared with the Naive Bayes model. This model is also beneficial because it works well with smaller data sets compared to a neural network which tend to need more data to be effective. Similar to the Naive Bayes model, this uses a bag of words framework as we apply word n-grams as features. While the Naive Bayes and SVM models are not a state-of-the-art approaches for sentiment analysis, it is important to explore and understand how it works on a sentiment analysis task to compare it to other models.

In our experiment, we only applied linear kernel function. Others such as radial basis kernel increase the running time while not significantly improving the results.

8.3 Random Forest

The Random Forest algorithm is a decision tree classifier and also referred to as en ensemble learning method. This tree based method produces many random decision trees where each individual tree predicts a class, and the class with the majority votes wins. The is an ensemble method in the way that it aggregates all of the total smaller votes together from each trees; this can also be described as a divide and conquer approach. One advantage of this is that it's clear what decisions the model is making whereas neutral networks can be more opaque.

We decided to use this model because it is very accurate and corrects itself to avoid overfitting. It is also a non linear model. Unlike the SVM and Naive Bayes algorithm, there is less control over model and not as amenable to linguistic insight. However, for this model we used bag-of-words approaches for features. In particular we utilize a dictionary approach that maps words with specific features relevant to sentiment. For example, our decision tree used the Harvard Inquirer lexicon to map various categories for words, including ‘Positiv’, ‘Negativ’, ‘IAV’, and ‘strong’. In this way, we create a feature vector of words and their labels for each review. We also used WordNet as a mapping tool to create synset vectors that represented the reviews.

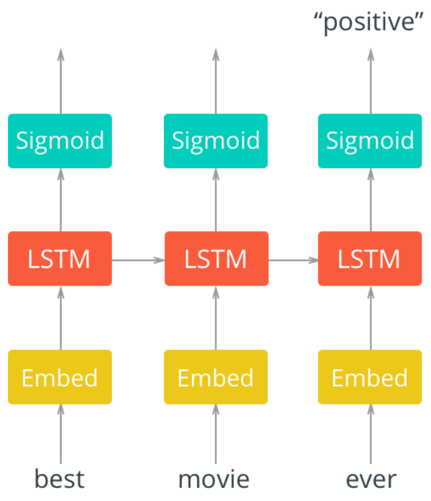
8.4 XGBoost

XGBoost is a gradient boosting decision tree algorithm similar to random forest. Gradient boosting is an ensemble method where trees are constructed iteratively and the model correcta errors that previous trees make, updating model with gradient descent.

In terms of providing linguistic insight, this model doesn’t allow for as much control as with the SVM and Naive Bayes model. In particular it doesn't allow for extensive feature engineering. Nevertheless this is a fast tool for learning better models and popular in Kaggle competitions.

8.5 RNN-LSTM

Recurrent Neural Network (RNN) is an extension of conventional feed-forward neural network. However, standard RNN has the gradient vanishing or exploding problems. In order to overcome the issues, Long Short-term Memory network (LSTM) was developed and achieved superior performance (Hochreiter and Schmidhuber, 1997). In the LSTM architecture, there are three gates and a cell memory state. Figure 6 illustrates the architecture of a standard LSTM[13].



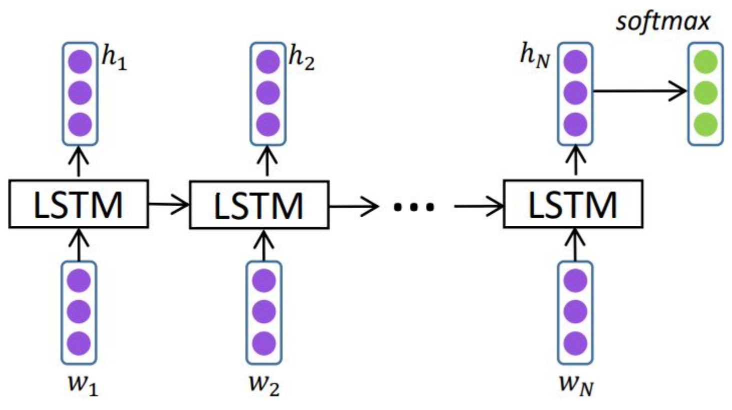


Fig.5. (Left) Model Structure for Word Sequence.
Fig 6. (Right) Architecture of a standard LSTM.

[13] Y. Wang et al., *Attention-based LSTM for Aspect-level Sentiment Classification*, Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 606–615.

According to Picture above, for all the RNN-LSTM model, it could be represented as the Embedding Layer, LSTM Layer with Dense, and Activation Functions for each Layer. Also, Inside the LSTM layer, LSTM will receive the word vector in a sentence whose length is N and h is the hidden vector.

9 Appendix 2 - Main Experiment Record

DataSet	Eval	CV	Test Acc.	Notes
8530 Sentences	valid-mlogloss:1.06511 train-mlogloss:1.05428	Not Yet	53.11%	XGBoost_model_eta_0.002_round_100_NumberFEeatures_3_TestSize_0.2_timestamp_2018_04_05_17_43
8530 Sentences	valid-mlogloss:0.984096 train-mlogloss:0.875597	Not Yet	53.52%	XGBoost_model_eta_0.002_round_2000_NumberFeatures_3_TestSize_0.2_timestamp_2018_04_05_17_45
8530 Sentences	7529/7529 loss: 1.0163 - acc: 0.5091 - val_loss: 0.9384 - val_acc: 0.5880	Not Yet	58.80%	LSTM_model_eta_0.0001_batch_32_epochs_1_layers_3_Embedding_keras.layers.Embedding_NumberTest_1000_timestamp_2018_04_05_18_00
8530 Sentences	7529/7529 loss: 0.1010 - acc: 0.9667 - val_loss: 1.8323 - val_acc: 0.4740	Not Yet	47.40%	LSTM_model_eta_0.0001_batch_32_epochs_10_layers_3_Embedding_keras.layers.Embedding_NumberTest_1000_timestamp_2018_04_05_18_05
Cleaned 8530 Sentences	7529/7529 loss: 0.7307 - acc: 0.7018 - val_loss: 0.8960 - val_acc: 0.6120	Not Yet	61.20%	LSTM_model_eta_0.0001_batch_32_epochs_2_layers_3_Embedding_keras.layers.Embedding_NumberTest_1000_timestamp_2018_04_06_15_43
Cleaned 8530 Sentences	valid-mlogloss:0.993448 train-mlogloss:0.905567	Not Yet	54.10%	XGBoost_model_eta_0.002_round_2000_NumberFeatures_2_TestSize_0.2_timestamp_2018_04_06_15_46

Cleaned 8530 Sentences	None	Not Yet	48.7%	MODEL: Bigram Multinomial Naive Bayes
				Precision = [0.53191489 0.24390244 0.48653345]
				Recall = [0.382263 0.06097561 0.77669903]
				F1 = [0.44483986 0.09756098 0.5982906]
				Accuracy = 0.4873752201996477

Cleaned 8530 Sentences	None	No t Yet	46.15%	MODEL: Bigram MultiClass SVM
				
				Precision = [0.44218316 0.23364486 0.54951456]
				Recall = [0.7123696 0.07225434 0.41253644]
				F1 = [0.5456621 0.11037528 0.47127394]
				Accuracy = 46.15%

Cleaned 8530 Sentences	None	No t Yet	62.71%	Unigram-Bigram Naive Bayes Model
				
				Precision = [0.63500678 0.24 0.63124336]
				Recall = [0.70164918 0.01923077 0.82044199]
				F1 = [0.66666667 0.03560831 0.71351351]
				Accuracy = 62.71%

Cleaned 8530 Sentences	None	No t Yet	61-64%	Unigram-Trigram Multinomial Naive Bayes
				
				Precision = [0.65803815 0.36363636 0.60823654]
				Recall = [0.71134021 0.02380952 0.8372093]
				F1 = [0.6836518 0.04469274 0.70458716]

Cleaned 8530 Sentences	None	No t Yet	58-59%	Unigram-Trigram MultiClass SVM
				
				Precision = [0.61815754 0.21461187 0.66122449]
				Recall = [0.68694362 0.15409836 0.67127072]
				F1 = [0.65073788 0.17938931 0.66620973]
				Accuracy = 58.49%

Cleaned 8530 Sentences	valid-mlogloss:1.00092
	train-mlogloss:0.969116
		No t Yet	51.79%	XGBoost_model_eta_0.002_round_2000_NumberFeatures_Unigram-Trigram Model_TestSize_0.2_timestamp_2018_04_09_16_01
				Precision = [0.60294118 0.33333333 0.48529412]
				Recall = [0.4239291 0.00320513 0.83193277]
				F1 = [0.49783174 0.00634921 0.6130031]
				Accuracy = 51.79%

Cleaned	loss: 0.7555 - acc: 0.6863	No	60.12%	LSTM_model_eta_0.0001_batch_32_epochs_

8530 Sentences	- val_loss: 0.9189 - val_acc: 0.6012	t Yet		2_layers_3_Embedding_keras.layers.Embedding_NumberTest_1705_timestamp_2018_04_09_16_09

Uncleaned Full Dataset	None	No t Yet	67.01%	Unigram-Trigram Multinomial Naive Bayes
				
				Precision = [0.5891905 0.73583789 0.64674719]
				Recall = [0.7072 0.62398094 0.72744607]
				F1 = [0.64282409 0.67530881 0.68472713]
				Accuracy = 67.01%

Cleaned 8530 Sentences	None	No t Yet	42.09%	Major Class Baseline

Cleaned 8530 Sentences	None	62. 4%	Not Yet	Performing feature selection based on chi2 independence test
				Training SVM
				Optimized parameters: LinearSVC(C=10.0, class_weight=None, dual=True, fit_intercept=True,
				intercept_scaling=1, loss='squared_hinge', max_iter=1000,
				multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
				verbose=0)

Cleaned 8530 Sentences	None	No t Yet	Not Yet	Performing feature selection based on chi2 independence test Training Random Forest OOB Score = 0.5170594442490327
Uncleaned Full Dataset	None	71. 2%	0.7368	SVM
Uncleaned Full Dataset	None	70. 6%	0.7249	Random Forest
Uncleaned Full Dataset	None	No t Yet	0.7273	Ensemble LSTM
Uncleaned Full Dataset	None	68. 32 %	0.7067	LSTM