Sentiment Analysis on Amazon Reviews

Connor Galvin
EECE 5644 Final Project
Northeastern University

1) Introduction

Sentiment analysis is the application of machine learning algorithms to decipher human emotions. There are many barriers to algorithmic learning of language. Sentiment is subjective, and humans disagree with machines and even each another on the tone of written text. The English language is also incredibly complex. Many words correspond to both positive and negative sentiment. Irony and sarcasm can be difficult to detect amongst people, and even harder to detect by machines. Another limitation of sentiment analysis is the large number of samples needed to gain insights. Conservative estimates place the number of words in the English language at 250,000 [1]. This means that tens of thousands of samples are needed to detect patterns in samples and quantify their sentiment.

By extracting individual words and their neighbors from text and vectorizing them by features such as count and Term Frequency Inverse Document Frequency (TFIDF), it is possible to train models to predict sentiment. By training these models on labeled samples, it is possible to classify new samples inputted to the model. The model finds differences in patterns of the words used in samples with positive and negative samples and uses this information to determine which class a new sample should belong to.

Sentiment analysis is particularly interesting to study because of its many applications. The huge amount of human written data on social media, internet forums, and product pages can be mined to gain insights into how people feel about particular events and media. I will be studying the application of sentiment analysis to user reviews on the largest eCommerce website in the United Sates, Amazon [2].

2) Related Work

In 2016, Kharde and Sonawane proposed a model to analyze sentiment of individual tweets on the social media platform, Twitter [3]. They outlined the basic principles of sentiment analysis, and experimented with different data preprocessing, feature extraction, and classification techniques. One big takeaway from their work is that the source the data is drawn from and the methods used for preprocessing the data are crucial to the performance of a sentiment analysis model. They witnessed more than 50% improvement in classifying tweets by tweaking how the data was preprocessed and how featured were selected for the model.

Sentiment Analysis is new and incredibly fast-growing field. A paper by Mäntylä, Graziotin, and Kuutila found that as of May 2017, over 99% of existing academic papers on the topic of sentiment analysis have been published after 2004 [4]. They also discovered a recent shift in the focus of sentiment analysis from online product reviews to social media, economics, cyberbullying, and political events. Another one of their findings was a shift in focus from classic algorithms like Naïve Bayes and SVM to more complex approaches, including neural networks and deep learning.

3) Methods

3.1 Obtaining data

The data used for this analysis is located on Kaggle.com [5]. The dataset is a CSV file containing 100,000 reviews taken from Amazon's website in 2014. Each review contains a Rating of 1, corresponding to one and two-star reviews, or a Rating of 2, corresponding to four and five-star reviews. The reviews come from a variety of different products in different shopping categories. All reviews are written in English.

3.2 Pre-processing data

Before working with the data, I ensured that it was in a format suitable for use in the model. With the use of Python, I removed all non-letter characters, punctuation, and capitalization of each word. This converted the data into an all lower-case series of words composed of the 26 letters of the standard English alphabet.

Another key component in sentiment analysis is filtering stop words from the data. A stop word is a commonly used word that is used for language structure but does not convey any new information [6]. Examples of stop words include: “the”, “and”, “or”, and “by”. It is important to filter these words out because they create noise in the data and lessen the effect of sentiment conveying words. I was able to filter out these stop words with the Natural Language Toolkit (NLTK) Python package [7].

3.3 Feature Extraction

The simplest way of performing sentiment analysis on text is by using unigrams. Unigrams represent each word in a review independently, without considering the influence of its neighbors. A unigram decomposition of the review “The book is not great” would read (the, book, is, not, great). [8]

Taking into account nearby words is a useful tool in sentiment analysis. For example, “great” conveys positive sentiment but “not great” conveys negative sentiment. For this reason, including bigrams and trigrams in sentiment models can often improve accuracy. Bigrams include a word’s nearest neighbor, while trigrams include two neighbors. A bigram decomposition of the review “The book is not great” would read [(the, book),(book, is),(is, not),(not, great)]. A trigram decomposition would read [(the, book, is), (book, is, not), (is, not, great)]. [8]

In order to convert my data into a format suitable for computation, I needed to vectorize it. The simplest way of doing this is to create a dictionary of all words used in the entire corpus of reviews. An n x m matrix is then created, where n is the number of reviews in the corpus, and m is the number of unique words in the corpus. The entries in the rows correspond to the number of times that a word in the corpus dictionary occurs in each review. This method is called the Count Vectorizer and I used Python’s Scikit-learn package to implement it. [9]

Another method for vectorization is with a Term Frequency Inverse Document Frequency (TFIDF) score. The TFIDF score weighs how important each word is by counting each word in a review and scaling by how many reviews in the corpus of

reviews contain that word. I implemented this method with Python's Scikit-learn package. See below for the formula used in calculating this metric. [9]

$$Tfidf_{i,j} = tf_{i,j} \times log(\frac{N}{df_i})$$

$Tf_{i,j}$ = total occurrences of word i in review j

df_i = total number of reviews containing i

N= Total number of reviews

3.4 Classification Algorithms

In order to actually make sense of the data, a classification algorithm must be used. I wanted to compare my results across different classifiers to see which would perform the best. To select which algorithms to use, I consulted Sci-kit learn's "Choosing the Right Estimator" chart [10]. I utilized the Sci-kit learn package to implement all of these algorithms.

Naïve Bayes

Naïve Bayes assumes that the value of a particular feature is independent of the value of any other feature. It is very simple, yet quite effective, making it a great baseline to compare other classifiers to. In the case of my experiment, Naïve Bayes does not take into account the association between the words contained in every review. [11]

K Nearest Neighbors Classifier

Under the K-neighbors classifier, an object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its *k* nearest neighbors. Essential it is classifying a sample based on the classes of other samples with similar features. In the case of my experiment, a newly seen review is compared to the 15 training reviews with the most similar words, and the class is decided as the more prevalent class in this sample (negative or positive). [11]

Linear SVC

Linear Support Vector Classification constructs a hyperplane or set of hyperplanes in a high space, which can be used to distinguish between classes. In the case of my

experiment, a linear hyperplane is drawn between the positive and negative class in the feature space (vectorization of words) with as large of a margin as possible in between. [11]

SGD

Stochastic Gradient Descent tries to classify by finding minima or maxima by iteration. It assumes a loss function, in my case this was Linear SVC, and it attempts to minimize by iteration. SGD can also be used in conjunction with logistic regression and perceptron. [11]

4) Results

The main metric that I used in evaluating my model is accuracy, the percentage of testing reviews whose true class was correctly predicted by the model. In the case of my experiment this is the number of positive reviews classified as positive plus the number of negative reviews classified as negative, divided by the total number of testing reviews.

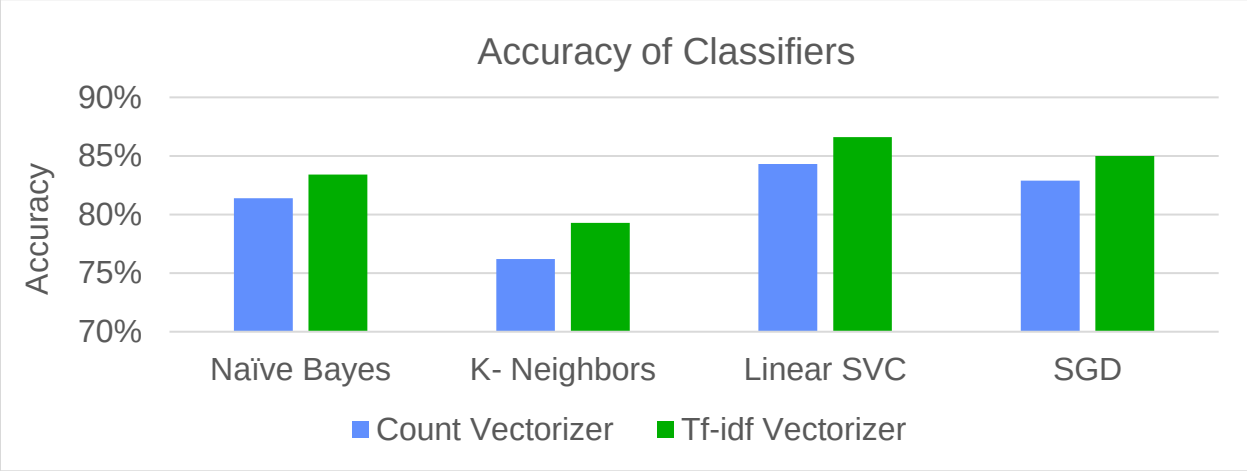

Figure 1- Percentage of negative and positive reviews classified correctly with 100,000 train/test samples, including unigrams, top 20% bigrams, and top 10% trigrams. Comparison across classifiers and vectorizers

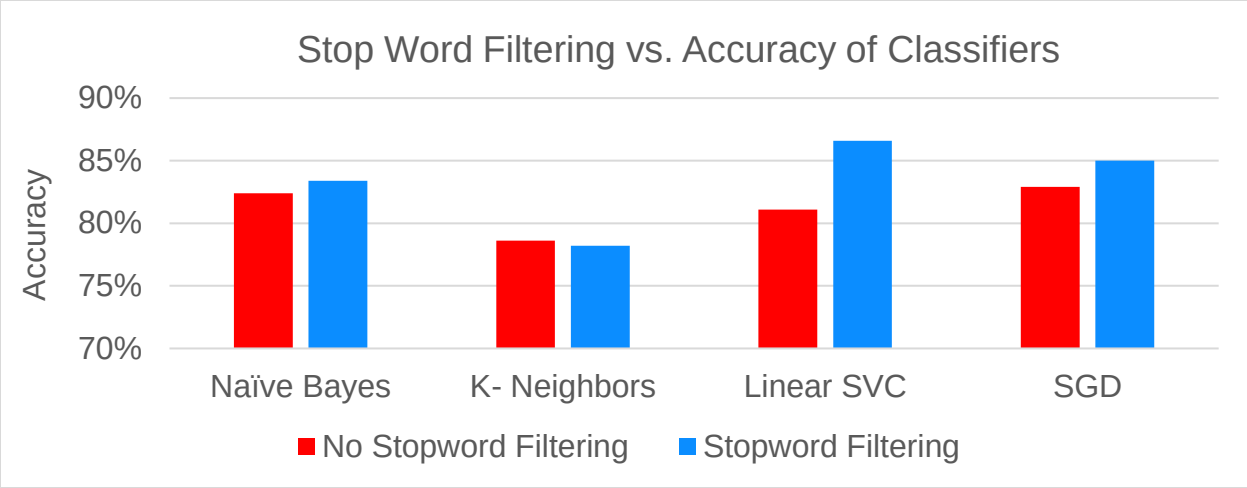

Figure 2- Percentage of negative and positive reviews classified correctly with 100,000 train/test samples, including unigrams, top 20% bigrams, and top 10% trigrams. Comparison across classifiers and stop word filtering

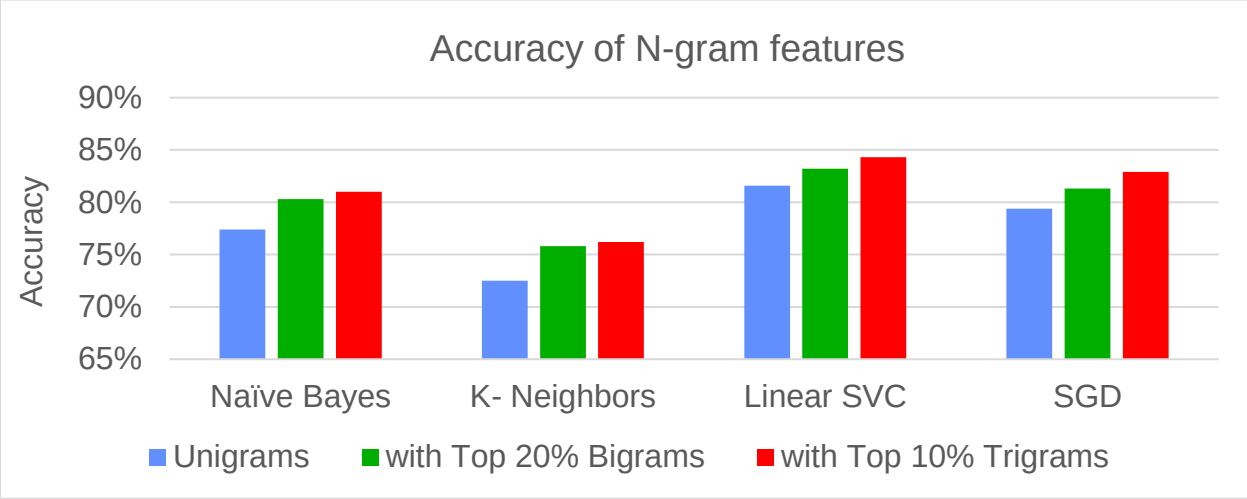

Figure 3- Percentage of negative and positive reviews classified correctly with 100,00 train/test samples, using count vectorization. The Bigram class includes all unigrams plus the top 20 % most relevant bigrams. The Trigram class includes the top 10% most relevant trigrams on top of the bigram class.

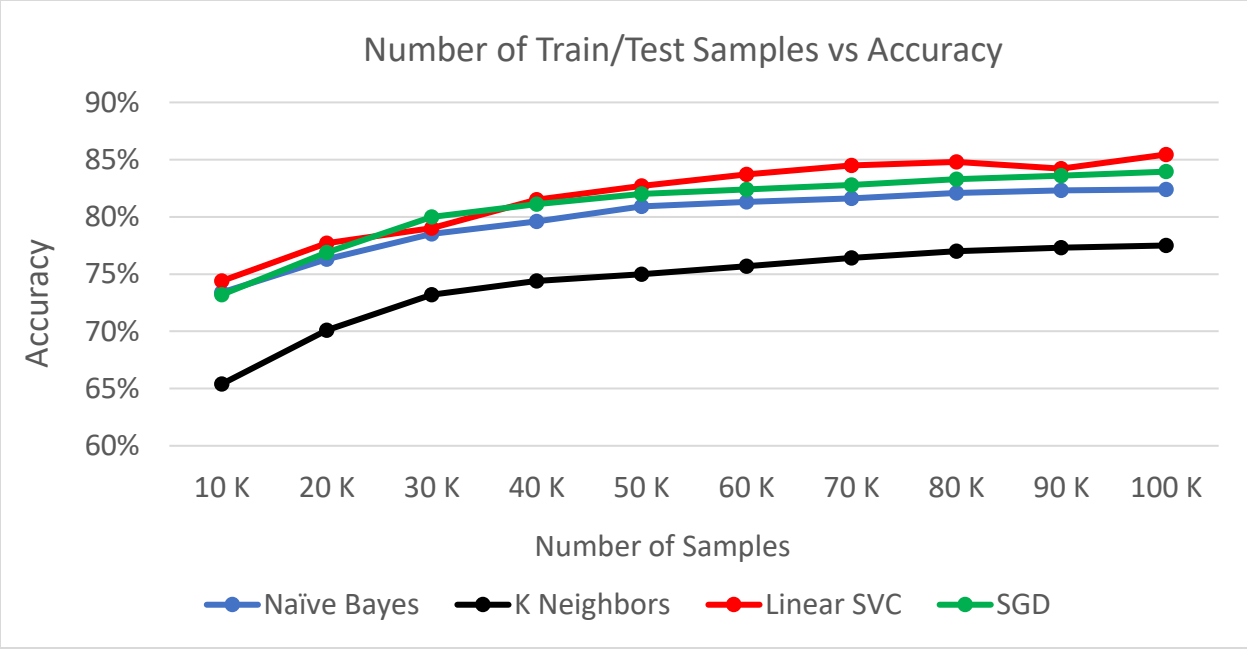

Figure 4- Percentage of negative and positive reviews classified correctly using count vectorization and a combination of uni, bi, and trigrams. A train/test split of 80/20 was used for each data point (For the first data point 8,000 reviews used for training, 2,000 for testing)

5) Conclusion

My results were straightforward and easy to follow. The highest accuracy I achieved between positive and negative reviews is 86.6 % using training/testing on 100,000 samples, stop word filtering, a combination of uni, bi, and trigrams, Tf-idf vectorizer, and Linear SVC. I made the following observations about how different variables effect my classification results:

- **Training Samples**- The more training samples, the higher the accuracy
- **Vectorization-** Tf-idf vectorizer performed marginally better than count vectorizer
- **Classifiers-** Linear SVC classifier performed the best, followed by SGD, Naïve Bayes, and KNN
- **N-gram features**- Including bigrams and trigrams increased accuracy but only when including the most relevant ones. By using a metric known as n-gram chi-

squared association feature, I was able to determine the most relevant bigrams and trigrams. Trial and error revealed that the best accuracy was a combination of all unigrams, the top 20% (by association measure) of bigrams, and the top 10% of trigrams. [12]

Interpretation of Results

The best classifier in my experiment correctly predicted the class of 86.6% of my testing reviews. I was quite impressed with results as it is markedly better than random guessing (50% accuracy). To get an idea of how well humans could predict these classes I tested my judgement on predicting the sentiment of 50 reviews and ended up correctly predicting 88% of them. It seems that my best classifier worked at about the same level as human intuition, with much faster results. This general sentiment analysis model could potentially be used for gaining large scale insights without sacrificing human time and effort.

Potential reasons for Error

Human language and its many parts are complex. Irony and sarcasm are frequently misinterpreted amongst humans and are very difficult to be detected by algorithms. Sentiment is subjective, relying on what humans determine to be true. This can cause problems because even humans disagree with each other. Additionally, the reviews were collected from products across multiple categories, leading to a large variation in language. This difference in word choice potentially caused difficulty in establishing patterns between negative and positive reviews.

Future Work

As an extension of my project, it would be interesting to take a deep learning approach to finding insights of the reviews using neural networks. Given a product, I would like to be able to analyze a large set of reviews to extract the top words or phrases that correspond to a negative review. This would allow a business to identify the worst features of a product that could benefit the most from improvement. Ultimately the goal for a large eCommerce company would be to automate the extraction of the most valuable insights that could lead to an improvement in sales and revenue.

References

1. “How Many Words Are There in the English Language” | Oxford Dictionaries.” Oxford Dictionaries | English, Oxford Dictionaries
2. “The Top 50 ECommerce Retailers in the US.” *Dallas Logistics & Warehouse*, www.darylfloodlogistics.com/the-top-50-ecommerce-retailers-in-the-us/.
3. A., Vishal, and Sc.Bs. Sonawane. “Sentiment Analysis of Twitter Data: A Survey of Techniques.” *International Journal of Computer Applications*, vol. 139, no. 11, 2016, pp. 5–15., doi:10.5120/ijca2016908625.
4. Mäntylä, Mika V., et al. “The Evolution of Sentiment Analysis—A Review of Research Topics, Venues, and Top Cited Papers.” *Computer Science Review*, vol. 27, 2018, pp. 16–32., doi:10.1016/j.cosrev.2017.10.002.
5. Bittlingmayer, Adam Mathias. *Amazon Reviews for Sentiment Analysis | Kaggle*, 24 May 2017, www.kaggle.com/bittlingmayer/amazonreviews.
6. “List of Stop Words.” *Yoast Knowledge Base*, kb.yoast.com/kb/list-stop-words/.
7. “Python Module Index.” *Python Module Index - NLTK 3.2.5 Documentation*, www.nltk.org/py-modindex.html.
8. “Kilian.thiel.” *KNIME*, www.knime.com/blog/sentiment-analysis-with-n-grams.
9. “API Reference.” *API Reference - Scikit-Learn 0.19.1 Documentation*, scikit-learn.org/stable/modules/classes.html#module-sklearn.feature_extraction.text.
10. “Choosing the Right Estimator.” *Choosing the Right Estimator - Scikit-Learn 0.19.1 Documentation*, scikit-learn.org/stable/tutorial/machine_learning_map/index.html.
11. “Supervised Learning - Classification and Regression.” *Structural Health Monitoring*, 2012, pp. 361–401., doi:10.1002/9781118443118.ch11.
12. “Source Code for Nltk.metrics.association.” Nltk.metrics.association - NLTK 3.2.5 Documentation, www.nltk.org/_modules/nltk/metrics/association.html