Comparing Algorithms for Sentiment Classification

Juweek Adolphe, Ressi Miranda, Zhaoyu Li, Shang Yi
University of Missouri – Colombia
Columbia, MO 65211
{ja4y5, rym4kd, zlht3, shangy}@mail.missouri.edu

Abstract

Sentimental Analysis has been a well explored field of classifying opinionated text into respective polarities (e.g. positive and negative) to gain an overall prescriptive of multiple opinions. Multiple algorithms have been used to classify text respectively to a polarity, especially with product reviews. Our goal is to improve data processing and feature selection to help improve the accuracy of Naive Bayes and Stochastic Gradient Descent (SGD) algorithms on book reviews. This project gets a best accuracy of 55%; we conclude there are no improvement from the usual standard.

1. Introduction.

The research field of sentiment analysis branches in areas of natural language processing and linguistics. It is a problem that has been worked on vigorously. Sentiment analysis is yet still a growing field of research study, and it doesn't take much imagination to understand why; opinion-rich resources such as online review sites and personal blogs are ubiquitous on the internet, and countless businesses, scientists, and other such entities are keen to the idea of tracking public sentiment from text, whether it be about particular products or public opinion of politician.

Given the massive amounts of product reviews that are now available to users and the amount the user needs to access is much large. Our work works on classifying new, unlabeled product reviews into either a positive or negative sentiment, focusing on improving the algorithm. Our project strips down a created classifier and divides the overall procedure into more detailed, expanded steps, for the overall aim is to discover any elements that can positively affect some characteristic of the classifier, whether it be time spent fitting the classifier to the feature vectors, memory usage, or the actual accuracy of the classifier. We mainly focused on increasing the classifier's accuracy. Our work is composed of three main parts; a type of feature vector model, an implementation - of lack of the implementation - of dimension reduction, and an actual classifier algorithm. We are using Naive Bayes, a commonly used classifier used for baseline classification analysis for our project in comparison with and stochastic gradient descent (SGD) [4].

2. Related Work.

Here we care to mention some of the related works regarding sentiment analysis. Sentiment Analysis has been explored thoroughly. Work has been done on feature selection [7, 24] lexicon builders [2, 7, 18], word polarity [15], supervised [4, 16], unsupervised algorithms [4, 11, 21], and other algorithms [19]. Approaches to sentiment analysis work on document level [19], sentence level [6, 7, 11, 26], and aspect-based features.

Different methodologies have been worked on grouping sentences together of similar attributes [21]. Sentiment Analysis has been explored in multiple domains with working with microblogs [1, 2, 7, 15, 23], product reviews, news and blogs [17], politics [13, 30] and other similar personal user domains. It has able been explored in different domains and domain adaptation of sentiment classification usage [5, 14]. Work has also been done on summarization of text using sentiment analysis [6, 7, 11]. Applications have been made to find correlations in people's actions and behavior [9, 12]. Even work has been done a larger scale, literature review has already been made [8].

There are several other areas of interest in sentiment analysis. Dealing context [2] and concepts. Summaries of current works, progress, and possible ideas have been made available [8, 10, 20].

Our work deals with being able to classify product reviews into polarity with the objective to increase two classifiers Naive Bayes and Support Vector Machine. Naive Bayes and Support Vector Machine are commonly used for sentiment analysis classification.

This paper is comparing Naive Bayes and a different algorithm Stochastic Gradient Descent.

3. Problem Formulation.

3.1. Problem Formulation.

As mentioned earlier, sentiment analysis is a fairly popular topic, and the demand for improvement of preexisting text classifiers - as well as the need for them to be accurate and scalable - has never been more prominent. Libraries such as scikit-learn have a plethora of different methods that can be used for text classification, but they are usually packaged into singular methods, so in-depth analysis and improvement can be somewhat difficult tasks. We aim to identify the variables of a classifier that are able to be improved upon in a significant manner, and allow the changes made in specific classifiers to be scalable and reusable. Our hypothesis is that there exists a specific variable that, when shown, can dramatically increase or decrease our accuracy.

3.1 Classification Process.

The first step of any classifier, independent of the scale, is to load our data and identify the categories that any new unprocessed text, once classified, will fall into. It is possible to pre-process the text; this is done to remove some of the noise that is inevitable going to be included with a corpus, as well as to format the text to make it easier for the classifier to read. By preprocessing test it reduces the number of dimensions in the feature vectors that will be created; this processing only picks out the *X* most important features and only includes them in our analysis. After we have loaded the dataset text and formatted it in an appropriate manner, we extracted features. We considered features to be words - and store them into feature vectors. Feature vectors, of course, act as a sort of lexicon, where each feature found in a labeled document is signaled to have that label based its presence or occurrences.

Defining which features are appropriate for analysis is fairly important. It is possible to use the unigram - also known as the “bag of words” approach - which simply takes every word as a feature and includes said word in the feature vector. There is also the bigram method, which takes combinations of two adjacent words rather than individual words, and, of course, a combination of both. After we have our created the feature vectors - each index of the vector, as mentioned before, having a particular label given in relation to its sentiment - we must train our classifier to perform categorization on new text by fitting it to the feature vectors we have present. We can do this with a multitude of different methods, though Naive Bayes is the one most preferred and there are two version of Naive Bayes that we can use: Multinomial Naive Bayes, which counts for word frequency and Bernoulli Naive Bayes accounts for the presence or absence for the word. Aside from Naive Bayes, we use stochastic gradient descent (SGD).

The first task of our project, was, of course, figuring out a way to strip down the procedure of text classification, and gain a full understanding of every layer present in the current text classification model. The idea of mixing and matching existing sci kit methods came into mind, as there are several different possible manners that the code for text classification can be written. The decision was made that any significant changes in accuracy would be because of an adjustment to our classifier; the only task at hand was finding out which specific adjustment would achieve the desired result of a higher accuracy.

3.3 Methods.

In order to gain a better understanding of the full formula of making a classifier, it was necessary to start on a somewhat small scale - one in which we would have a fair amount of control of a number of variables.

To implement the experiments, we utilize the python machine learning library known as scikit-learn. Scikit-learn allows users the option to make a Pipeline, which assembles several steps and condenses them into one object, allowing them to be fitted, transformed, and cross-validated together. For our experiment, each Pipeline was composed of three main parts; a type of feature vector model, an implementation - of lack of the implementation - of dimension reduction, and an actual classifier algorithm. In terms of dimension reduction, we choose to implement the chi-squared test or not, to remove more features from the data. For our comparison we are using Bernoulli Naive Bayes, Multinomial Naive Bayes, and a linear classifier with SGD learning. Scikit-learn has all of the mentioned methods and algorithms implemented in usually a few lines of code. We, early on, opted to use this library as opposed to writing our own classification method from scratch.

We used the Multi Domain Sentiment from Blitzer 2007[5] for the sentiment classification, we specifically used Amazon book reviews from the dataset for our work.

4. Implementation.

The four steps of building a classifier are, for the most part, implemented in methods provided by the scikit-learn library. As such, there is not much room for adjustment in algorithm implementation, there are usually numerous ways in which we can perform a specific step; an example was mentioned above, where the choice of storing features as unigrams or a combination of both can alter the overall frequency of a classifier. In order to improve a classifier of any kind, it was decided to first mix and match these different approaches and derive which methods are most efficient for their particular layer in our classifier. We hope to gauge whether feature extraction (unigram, bigram, chi-square test), feature vector (count vector, hash vector. TF-IDF), and algorithm (Naive Bayes, SGD) combination which would work the best. We choose a 10-Fold Cross Validation to validate our results.

4.1 Term Vector.

When scanning raw text and deciding which features will be added to the feature vector, there are a few word processing models that can be used. The most predominant in the field of text classification is unigrams. In our term vectors we used unigrams and a combination of unigram and bigram since our dataset used this scheme.

4.2 Feature Extraction.

For the feature vector models, a vector space that uses were a few options to choose from. One of the vectors models used was the CountVectorizer, which converts a collection of text documents into a matrix of token counts. The HashingVectorizer, an alternative, converts a collection of text documents to a matrix of token occurrences. Lastly, there was the TfidfVectorizer, which converts a collection of raw documents to a matrix in the form of term frequency versus inverse document frequency.

4.3 Feature Selection and Dimension Reduction.

As we will be creating a feature vector, the issue of size; larger datasets often correlate to an increase in accuracy, but an increase in the size of a dataset leads to a number of problems concerning storage capabilities, and processing speed. Also, if this classifier is to be scalable it is necessary to scale back the size of the feature vector, reducing the number of accounted features. The chi-squared test serves this function exactly; the test determines the significance of a word by using the word's frequencies in multiple documents, and our classifier can decide the percentage cut off for our features. For this experiment, when the chi-squared test is implemented, only the top 30% of features are added to the vector. We intend to compare the usage of the chi-square test and without.

4.4 Classification Algorithms.

4.4.1 Bernoulli Naïve Bayes. The Bernoulli Naïve Bayes takes account of the presence and absence of a feature. Posterior probabilities are based on the features presence or absence in the overall document. This technique is usually used for document level sentiment classification and sentence sentiment classification.

4.4.2 Multinomial Naïve Bayes. Another way to classify test is Multinomial Naïve Bayes. Similar approach as the Bernoulli, but the difference that this takes into account the feature frequency by finding the posterior probabilities hoping that each features as independence from other features.

4.4.2 Stochastic Gradient Descent (SGD). The classifier that we use is probably the most important part of our algorithm, and it was important to get a varied list of possible alternatives to the standard Multinomial Naive Bayes and Bernoulli Naïve Bayes. For our experiments, we included both the Multinomial and Bernoulli Naive Bayes classifiers. In addition, we have included a stochastic gradient descent.

5. Experimental Results.

For our experiment, we decided to record the accuracy, found using 10-fold cross-validation, of a classifier using a dataset of 200 book reviews, 1000 of which were positive, and 1000 of which were negative [5]. Our classifier was coded with the utilization of a Pipeline object in scikit-learn, which made small scale manipulation and exchanging methods in each layer a rather simple process.

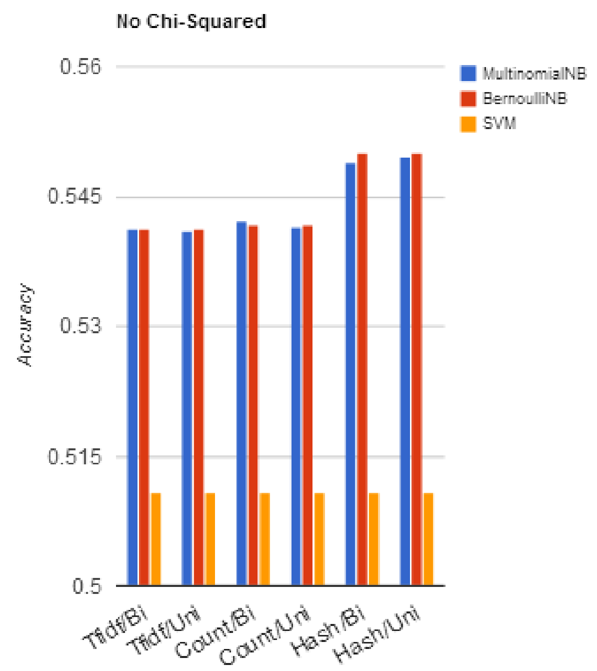

Figure 1. No features removed.

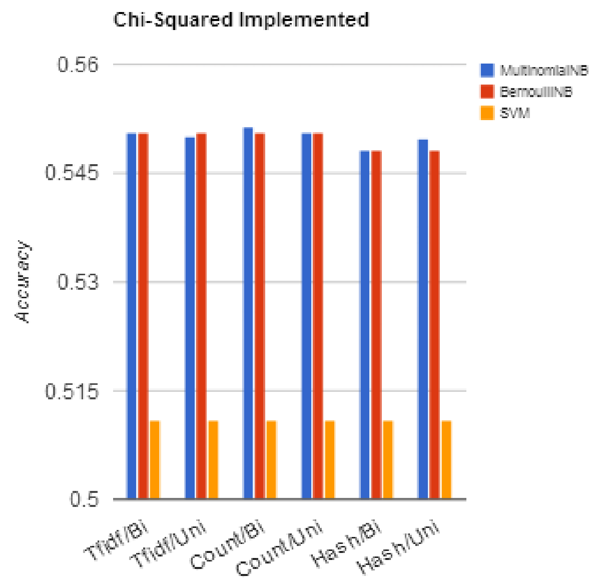

Figure 2. Featured removed.

Although the batch of reviews were the same, we still ran our experiments in two different ways: the first, figure 1, had the chi-squared test implemented while the second, figure 2, did not. Each graph has six collections on columns; two of them will use the same feature vectorizer process, though one will use unigrams while the other a combination of unigrams and bigram, represented by 'Bi'. Each collection has three columns - one for each classification algorithm. This results in a total of 18 different columns for every graph, and 36 total columns for both of the graphs.

Looking at the graphs, there are a few things that can be things that can be extrapolated from the graphs. The first, and almost disappointing, thing that can be said is the fact that there isn't a consistent correlation between any significant increase or decrease in accuracy and the use of unigrams and a combination of unigram and bigrams. This simply means that, however the user extracts the features from the raw text - whether as a combination or a unigram - the accuracy will not change too much. This is present in both the graph that has the chi-squared test implemented and the graph that does not.

The SGD Classifier that we used for this dataset is, surprisingly, much less accurate than the other two classification methods. With the chi-squared test implemented, the greatest accuracy came with the combination of extracting bigrams and unigrams from the raw data, implementing a CountVectorizer, and using the MultinomialNB algorithm for classification. Without the chi-squared test implemented, the highest accuracy came from extracting both unigrams and bigrams from the text, using a TD-IDF, and implementing the Multinomial algorithm for classification.

Although the numbers presented are utilized for now, we will need to double check the accuracy of this experiment by recreating it in a number of ways. For all of our tests, we simply used one data set - which had a peculiar formatting and was already pre-processed with unigram and bigrams, but to keep the experiment valid we treated bigrams as unigrams - and received the accuracies that we have shown above. However, it is possible that this data set, because of its odd formatting, is inaccurate. It is also possible that one of the two previously mentioned combinations received the highest accuracy by nothing more than chance, and, if we recreate this experiment, with a different dataset, the results might be different.

10. References

[1] González-Ibáñez, Roberto, Smaranda Muresan, and Nina Wacholder. "Identifying sarcasm in Twitter: a closer look." Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2. Association for Computational Linguistics, 2011.

[2] Whitelaw, Casey, Navendu Garg, and Shlomo Argamon. "Using appraisal groups for sentiment

analysis." Proceedings of the 14th ACM international conference on Information and knowledge management. ACM, 2005.

[3] Jiang, Long, et al. "Target-dependent twitter sentiment classification."Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1. Association for Computational Linguistics, 2011.

[4] Pang, Bo, Lillian Lee, and Shivakumar Vaithyanathan. "Thumbs up?: sentiment classification using machine learning techniques." Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10. Association for Computational Linguistics, 2002.

[5] Blitzer, John, Ryan McDonald, and Fernando Pereira. "Domain adaptation with structural correspondence learning." Proceedings of the 2006 conference on empirical methods in natural language processing. Association for Computational Linguistics, 2006.

[6] Zhu, Linhong, et al. "Graph-based informative-sentence selection for opinion summarization." Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining. ACM, 2013.

[7] Khuc, Vinh Ngoc, et al. "Graph-based informative-sentence selection for opinion summarization." Proceedings of the 27th Annual ACM Symposium on Applied Computing. ACM, 2012.

[8] Cambria, Erik, et al. "Big social data analysis." Big Data Computing (2013): 401-414.

[9] Kucuktunc, Onur, et al. "A large-scale sentiment analysis for Yahoo! answers."Proceedings of the fifth ACM international conference on Web search and data mining. ACM, 2012.

[10] Cambria, Erik, et al. "New avenues in opinion mining and sentiment analysis." (2013): 1-1.

[11] M. Hu and B. Liu, “Mining and summarizing customer reviews,” in Proceedings of the 10th ACM SIGKDD. New York, NY, USA: ACM, 2004, pp. 168–177.

[12] Garas, Antonios, et al. "Emotional persistence in online chatting communities."Scientific Reports 2 (2012).

[13] Tumasjan, Andranik, et al. "Predicting Elections with Twitter: What 140 Characters Reveal about Political Sentiment." ICWSM 10 (2010): 178-185.

[14] Glorot, Xavier, Antoine Bordes, and Yoshua Bengio. "Domain adaptation for large-scale sentiment classification: A deep learning approach." Proceedings of the 28th International Conference on Machine Learning (ICML-11). 2011.

[15] Hamdan, Hussam, Frederic Béchet, and Patrice Bellot. "Experiments with DBpedia, WordNet and SentiWordNet as resources for sentiment analysis in micro-blogging." Atlanta, Georgia, USA (2013): 455.

[16] Gamon, Michael. "Sentiment classification on customer feedback data: noisy data, large feature vectors, and the role of linguistic analysis." Proceedings of the 20th international conference on Computational Linguistics. Association for Computational Linguistics, 2004.

[17] Godbole, Namrata, Manja Srinivasaiah, and Steven Skiena. "Large-Scale Sentiment Analysis for News and Blogs." ICWSM 7 (2007).

[18] Wilson, Theresa, Janyce Wiebe, and Paul Hoffmann. "Recognizing contextual polarity: An exploration of features for phrase-level sentiment analysis."Computational linguistics 35.3 (2009): 399-433.

[19] Pang, Bo, and Lillian Lee. "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts." Proceedings of the 42nd annual meeting on Association for Computational Linguistics. Association for Computational Linguistics, 2004.

[20] Pang, Bo, and Lillian Lee. "Opinion mining and sentiment analysis."Foundations and trends in information retrieval 2.1-2 (2008): 1-135.

[21] Turney, Peter D. "Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews." Proceedings of the 40th annual meeting on association for computational linguistics. Association for Computational Linguistics, 2002.

[22] Dave, Kushal, Steve Lawrence, and David M. Pennock. "Mining the peanut gallery: Opinion extraction and semantic classification of product reviews."Proceedings of the 12th international conference on World Wide Web. ACM, 2003.

[23] Pak, Alexander, and Patrick Paroubek. "Twitter as a Corpus for Sentiment Analysis and Opinion Mining." LREC. 2010.

[24] Hatzivassiloglou, Vasileios, and Kathleen R. McKeown. "Predicting the semantic orientation of adjectives." Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics. Association for Computational Linguistics, 1997.

[25] Garcia, David, et al. "Political polarization and popularity in online participatory media: an integrated approach." Proceedings of the first edition workshop on Politics, elections and data. ACM, 2012.

[26] Liu, Yang, et al. "Sentence-Level Sentiment Analysis in the Presence of Modalities." *Computational Linguistics and Intelligent Text Processing*. Springer Berlin Heidelberg, 2014. 1-16.