Effective Sentiment Analysis using a Bag of Words Representation

Crystal Qian
Princeton University
cqian@princeton.edu

Lachlan Kermode
Princeton University
lkermode@princeton.edu

Abstract

Sentiment analysis is a fundamental manipulation in the hegemonic and technocratic cybernetic regime of our era: data must be classified! Language must be reduced to its sentiment, for identifying speech that does not accord with the current political norms and filtering it from the observable zone of discourse is high priority in our heydey. In being able to classify text without human observation, we are able to more successfully set strict parameters on 'legitimate' language, and at the same time conveniently absolve ourselves from the direct responsibility in the eyes the moralising majority. We use our sentiment analysis on 2 data sets, using a bag-of-words representation in both cases. The first data set we used is the prescribed set of 1385 tweets that are human-annotated with 0 for negative sentiment, and 1 for positive sentiment. We then apply these methods to a second annotated data set of "pros/cons" reviews, used in previous sentiment classification studies [6, 7].

1 Introduction

Our goal in this assignment was to experiment with various preprocessing and classification techniques to classify the sentiments of the given dataset with high accuracy. Additionally, we wanted to test on additional datasets to verify that the accuracy rates determined in our proposed algorithm are not the result of overfitting.

Our process takes two steps: first, we preprocess the data and prune a bag-of-words representation of each document in the dataset. Then, we use a variety of different classifiers (and bagging with a meta-classifier) tuned through an exhaustive grid search on hyper-parameters to obtain metrics. For each different classifier, we attempted to select features from the source data to put into the bag of words through a combination of research and intuition, validating our hypotheses against the data set.

We present an approach that yields approximately 97.8% accuracy on the given dataset.

2 Methods

The most successful model in our experience was a simple decision tree classifier, trained on data preprocessed in the following ways:
- Ignored word count threshold, and simply test for presence.
- Filter neutral words (our definition of neutrality is defined below).
- Modify words that follow negative STOPwords.
- Optimise hyper-parameters of select classifiers.

Accuracy	0.978
Precision	0.993
Sensitivity	0.963
Specificity	0.993
F1 Score	0.978
False Positives	2
False Negatives	11

In the following sections, we will discuss how we optimized this result by implementing features in preprocessing and classifications steps, using accuracy as the comparator and .978 as our baseline. Notice that decision tree classifiers do not output deterministic results; that is, the accuracy varies from run to run. So, the accuracy scores in the following tables are averages over 10 runs, and may not necessarily be consistent across tests. However, they are approximately around .978 to provide a sufficient comparison.

2.1 Word Count Threshold

The original preprocessing script uses a word count threshold; that is, a word is not placed into our vocabulary unless its frequency across documents surpasses the threshold. Better performance is typically achieved by accounting only for feature presence, not frequency [3] (thus setting word count threshold to 1). We found that accuracy was best with word count threshold set to 2 (vocabulary size 867 after preprocessing) when using the decision tree classifier; however, for other classifiers described, accuracy improved with word count threshold set to 1 (vocab size 1282 after preprocessing).

Word Count Threshold	Accuracy
1	0.958
2	0.97
3	0.953
4	0.942

2.2 Filtering Neutral Words

We filter out neutral words in two ways: first, we used an existing opinion lexicon with positive and negative words [1]. Words would only make it into the final vocabulary set if they existed in the positive or negative word list. Another approach assigned each word a neutrality score p/t, where p is its frequency in positively-classified documents, and t is its total frequency across all documents. Each score ranges from 0 to 1, where a score around .5 indicates total neutrality (equal presence in positive and negative word lists). Then, we would filter words out of our vocabulary if their neutrality score fell between our minimum and maximum neutrality thresholds.

No neutral filtering	Lexicon filtering
0.947	0.802

min.5,max.5	min.3,max.7	min.3,max.6	min.3,max.55	min.35,max.6
0.95	0.947	0.97	0.947	0.955

2.3 Negative Words

The original preprocess code filters stopwords, or, words that are fairly neutral such as ”me,” ”myself,” ”yours,” etc. These stopwords are determined by the nltk.corpus stopwords library. As shown in previous classification work [2, 3], adding ”-NOT” (or any other negative marker) after a negative word yielded better results. For example, ”not suggest” would turn into ”not suggest-NOT” if we did not additionally filter the negative word, or ”suggest-NOT” if we do filter the negative word. Thus we modify the stopword list to omit negative stopwords. Although decision trees benefit from filtering all stopwords (857 instead of 867 vocab size when not filtering), note that other classifiers performed better on the larger vocabulary size.

Filter all stopwords	No filter negative stopwords, Add -NOT	Filter all stopwords, add -NOT
0.923	0.965	0.97

2.4 Bag of Words Representation: Frequency, Presence, TD-IDF

The bag of words representation keeps a vector for each document; each value in the vector is a word in the vocabulary's frequency in the given document. We implemented TF-IDF (weighing each word count dependent on frequency in the document) as an alternative to word frequency. Although TF-IDF performs better on the multinomial classifier, frequency tends to perform better on decision tree classifier.

Presence	Frequency	TFIDF
0.978	0.978	0.97

2.5 Filtering numbers and ASCII

We expected filtering numbers (removing words containing numerical symbols) and non-ascii words (removing words including non-ascii symbols) to increase the accuracy rate. Filtering non-ascii had no effect on accuracy, perhaps because the only word filtered out was "clich". Surprisingly, filtering numbers resulted in a worse accuracy rate. The .002 error is attributed to one additional misclassified false negative: "I rate this movie 9/10." (344 in train.txt).

Filter numbers and non-ASCII	Filter numbers	Filter non-ASCII	No filter
0.973	0.973	0.978	0.978

3 Results

We experimented with a range of different classifiers:
- Adaboost with 25 estimators and no random state
- Bernoulli
- Decision Tree
- Gaussian
- Linear SVC
- Multinomial Naive Bayes, with alpha = .73 and fit prior
- Tuned RBF SVC, with C=1 and gamma = .001

Additionally, we implemented bagging with a voting classifier that took in Bernoulli, Multinomial and Decision Tree classifiers. These three were chosen for their good performance on the dataset. Based on previous works [2, 3, 4] and the nature of the data, we predicted that multinomial naive bayes and SVC would yield the highest accuracy. However, when we fine tuned pre-processing steps to each classifier, we noticed that decision trees actually yielded a higher maximum accuracy.

In order to tune hyper-parameters for adaboost, multinomial, and SVC, we tested the classifier using a range of different hyperparameters through an exhaustive grid search, starting with a net across exponential values (1,10,1000), and then subsequently searching in the domain of the most successful.

The following table shows classifier scores across a consistent implementation of the preprocessing and filtering decisions that were created to optimize decision trees. Note that we can fine-tune preprocessing decisions to optimize scores for different classifiers, but we choose to use the best version for decision trees because it had the highest accuracy rates across the board.

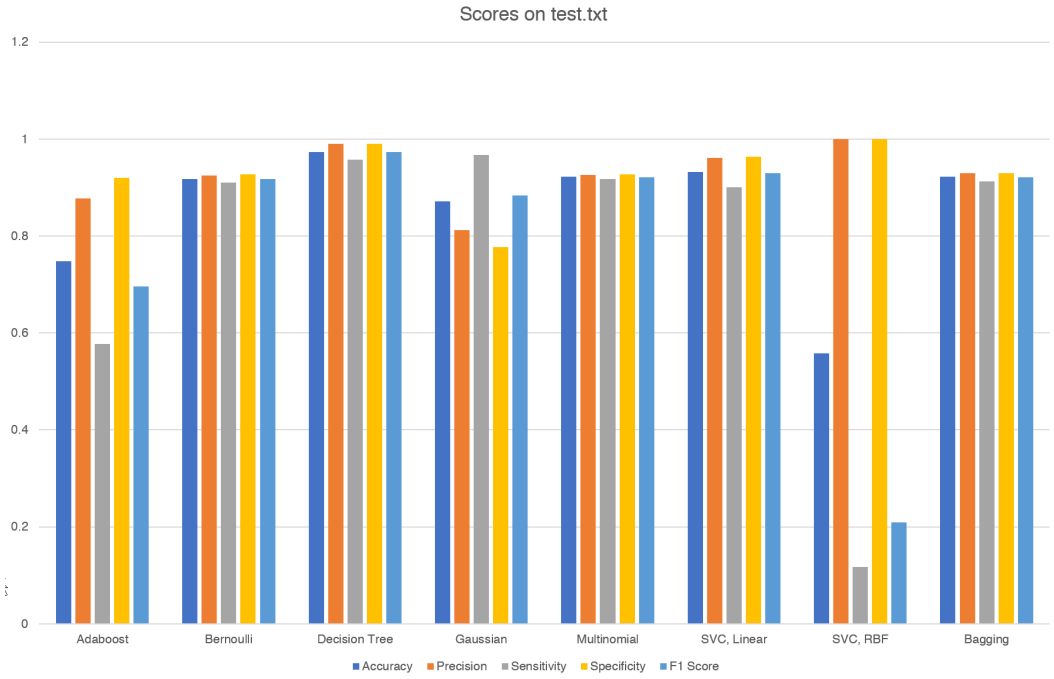

Figure 1: Graph of metrics across a range of different classifiers.

Name	Accuracy	Precision	Sensitivity	Specificity	F1
Adaboost	0.748	0.878	0.577	0.92	0.696
Bernoulli	0.918	0.925	0.91	0.927	0.918
Decision Tree	0.973	0.99	0.957	0.99	0.973
Gaussian	0.872	0.812	0.967	0.777	0.883
Multinomial	0.922	0.926	0.917	0.927	0.921
SVC, Linear	0.932	0.961	0.9	0.963	0.929
SVC, RBF	0.558	1	0.117	1	0.209
Bagging	0.922	0.929	0.913	0.93	0.921

4 Additional Dataset

We used an existing pro/cons dataset [6, 7] used in previous sentiment analysis research to verify the robustness of our existing approach. This data was significantly larger than the given train.txt and test.txt, with a combined 45,878 documents. We split the data into training and test sets through cross validation on 10 folds. The following results are the averages of these results across 100 iterations:

Name	Accuracy	Precision	Sensitivity	Specificity	F1
Adaboost	0.856	0.92	0.777	0.934	0.842
Bernoulli	0.936	0.943	0.926	0.945	0.934
Decision Tree	0.969	0.977	0.959	0.978	0.968
Gaussian	0.753	0.668	0.993	0.519	0.799
Multinomial	0.93	0.936	0.931	0.938	0.934
SVC, Linear	0.941	0.948	0.931	0.951	0.94

The SVC with RBF kernel timed out after half an hour. However, notice that our scores across the two datasets are consistent, suggesting that results on the initial dataset are not due to overfitting.

5 Discussion and Conclusion

We approached the problem by laying out a series of different pre-processing decisions, selecting a broad range of classification methods, optimizing each set of pre-processing decisions to each classification method, and selecting the set with the highest accuracy result (in this case, decision tree classification with the decisions specified above) to compare across the board.

Our pre-processing decisions were roughly similar to those introduced in previous works; however, we offer a new approach that uses a decision tree classifier instead of the SVC/Multinomial Naive Bayes classifiers that are popularly used. Because we used this new classifier, parameters to the pre-processing decisions show varying results. (For example, when setting word count threshold to 1 in multinomial and SVC classification, we produced a higher accuracy rate as consistent with [2][4]. However, this word count threshold set to 2 provided higher accuracy in decision trees.) We learned that pre-processing decisions do not yield consistent results across all classifiers, which intuitively can be explained by the different approaches classifiers take in making predictions.

It's difficult to reason exactly why a decision tree classifier performs better than the range of SVC models or a Bayes approach. One hypothesis is that the data we are working with are short, declarative sentences, rather than more nuanced dissertations. They therefore tend towards 'patterns' of positive and negative, which are well represented by a decision tree (e.g., sentence contains 'good' and no negatives, probably a positive review). However, this hypothesis could greatly benefit from further analysis of the data and results.

In all, our .978 accuracy rate on the given training set using a decision tree classifier suggests that we should further explore the use of different classification systems in sentiment classification. Additionally, pre-processing results shown to be successful for one classifier should not be generalized across all classifiers.

Acknowledgments

5.1 References

1. Hu, Minqing, and Liu, Bing. "Mining and Summarizing Customer Reviews." Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2004), Aug 22-25, 2004, Seattle, Washington, USA.
2. Pang, Bo, and Shivakumar Vaithyanathan. "Thumbs Up? Sentiment Classification Using Machine Learning Techniques." International Journal of Science and Research (IJSR) 5.4 (2016): 819-21. Web.
3. Pang, Bo, and Lillian Lee. "Opinion mining and sentiment analysis." Foundations and Trends in Information Retrieval 2.1?2 (2008): 1-135.
4. Go, Alec, Richa Bhayani, and Lei Huang. "Twitter sentiment classification using distant supervision." CS224N Project Report, Stanford 1.12 (2009).
5. Wang, Sida, and Christopher D. Manning. "Baselines and bigrams: Simple, good sentiment and topic classification." Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2. Association for Computational Linguistics, 2012.
6. Ganapathibhotla, Murthy, and Bing Liu. "Mining opinions in comparative sentences." Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1. Association for Computational Linguistics, 2008.
7. Liu, Bing, Minqing Hu, and Junsheng Cheng. "Opinion observer: analyzing and comparing opinions on the web." Proceedings of the 14th international conference on World Wide Web. ACM, 2005.