Colloquial Tomatoes

Darren Edmonds
dedmond1@uci.edu

Karthik Gajulapalli
kgajulap@uci.edu

Brooke Ryan
brooke.ryan@uci.edu

Abstract

In this project, we aim to build a machine learning model that successfully classifies the sentiment of IMDB movie reviews. In the sentiment analysis problem space, utilizing a Naive Bayes model is a popular approach, due to its relatively fast computation times and accurate results on balanced data sets. Despite its popularity, Naive Bayes is far from a perfect classifier; its strong assumption of conditional independence removes some of the nuance that is inherent in natural language structure. In this project, we will attempt to use ensemble methods to boost classification performance, leveraging multiple classifiers to improve overall confidence regarding model predictions. Our main challenge is to overcome multiple alternative learning algorithms or variations to Naive Bayes, as well as coming up with a method to weigh the votes between models. Our primary goal with this project is to learn more about methods for text classification in the scope of the field of Machine Learning – we seek to understand and implement the standard as well as experimental approaches, in order to foster a holistic understanding of state-of-the-art sentiment analysis.

1 Introduction

A portmanteau of "colloquial vernacular" and movie review website "Rotten Tomatoes", an aptly-chosen title for the project given its language processing subject, we survey multiple machine learning techniques in binary sentiment classification. We experiment on the IMDB Movie Reviews dataset [1], with the ultimate goal of gleaning insight on the best machine learning models for natural language processing. Within the sentiment analysis problem space, utilizing a Naive Bayes model is a standard approach, given its relatively fast computation times and accurate results on balanced data sets. Despite those advantages, Naive Bayes is far from an ideal classifier, as its strong assumption of conditional independence removes some of the nuance that is inherent in natural language structure. In this project, we compare a variety of classifiers against the standard approach to improve overall confidence regarding model predictions. Additionally, we examine whether applying ensemble methods further enhances classification performance. Our main challenge is to overcome multiple alternative learning algorithms or variations to Naive Bayes, as well as coming up with a method to weigh votes between models. Our primary goal with this project is to learn more about methods for text classification in the scope of natural language processing – we seek to understand and implement standard as well as experimental approaches, in order to foster a holistic understanding of state-of-the-art sentiment analysis. Our experiments span a variety of machine learning techniques; ultimately, our most successful model was a Logistic Regression model, followed by Support Vector Machines. Our results imply that Naive Bayes models are not necessarily better than other common machine learning models for text-based sentiment analysis. Additionally, our results suggest that a neural network model with a sigmoidal activation function might result in even higher accuracy, given the favorable results seen with simple logistic regression. Ultimately, our experiments further validate the promise of deep learning within the natural language processing space.

UC Irvine Machine Learning (CS 273), Winter Quarter 2021

2 Problem Statement

2.1 Main Goal

A major area of interest for the field machine learning is the accurate prediction of sentiment from text. Currently, it is difficult to do so as sentiment is often nuanced, and even when it is not so, it can be difficult for algorithms to parse through syntactic noise and colloquial vernacular. Thus, the main issue our project tries to address is how to best leverage current machine learning algorithms to predict sentiment in text with the highest possible accuracy.

2.2 Dataset Utilized

Our project utilizes the popular Large Movie Review Dataset [1]. Originally provided by Maas et al. in 2011, this dataset has become a benchmark in a variety of machine learning research tasks. The dataset contains approximately 50,000 highly polar movie reviews provided by IMDB users, lending itself well for binary classification. Each review is labeled as either positive or negative and the data is split in half between a training and testing set. While the review data can be thought of as qualitative, we decided to implement text preprocessing models to transform words into machine-readable, quantitative feature vectors.

3 Methods

3.1 Dataset Preprocessing

The IMDB reviews were originally aggregated from a webscraper, so in order to ensure classification models can be compared fairly, preprocessing was a necessary step. All special characters and capitalization were removed, and each review was ensured to comprise solely of a collection of English words.

While the use of stemming is very popular in the natural language processing literature, the authors did not observe any realizable difference between implementing stemming and going without. We believe that this is relatively surprising, as stemming is known to be useful when gaining some semantic information from a sentence (intuitively, it would help group words that mean the same thing). This could be explained by parts of words such as tense being slightly important with regards to sentiment prediction, resulting in an equivalent loss of value to the expected gain when implementing stemming on the review data.

3.2 Models

To achieve our goal of training a classifier that predicts sentiment of reviews on the test set, we ran variations of the following classifiers: Naive Bayes, Logistic Regression, Decision Trees, Random Forest, and Support Vector Machines. For the Naive Bayes classifiers, we experimented with Bernoulli and Multinomial Naive Bayes; additionally, we experimented with both a Count and a TFIDF vectorizer for the Multinomial Naive Bayes model (for the Bernoulli model, the vectorizers are equivalent as features are shown as binary, so we only count presence of a word once). We implemented three variants of Decision Trees (classic, boosting, and bagging), as well as both a classification and a regression based Random Forest model. We also implemented some ensemble methods over our classifiers, as we believed that combining the predictions of our top models could lead to further enhanced performance. All our code can be found on github: `github.com/brookekelseyryan/ColloquialTomatoes`

4 Experiments

4.1 Feature Construction

The next step was to define a good feature set for our classifiers. We used two main approaches in constructing a feature set: a vectorizer-based approach, and a lexicon-based approach. A vectorizer takes as input a collection of sentences and generates a feature set; thus, each

Figure 1: Word cloud visualization of positive and negative reviews on the left and right respectively.

word is assigned some weight based on frequency. Although we tried a count vectorizer for our Naive Bayes models, we decided to use TFIDF vectorizers as the gold standard for our models given the fact that the vectorizer takes into account document frequency of words (which can be thought of as relative value of words) in addition to frequency within a review. The Lexicon-based approach was instead based on the NRC Emotion Lexicon [2], which contained associations of 14,000 words with 8 core emotions as well as positive and negative sentiment. Using this lexicon, we enumerated emotional associations for each word in a movie review and thus constructed a feature vector of length 10, with each index corresponding to an emotion or sentiment. We ran our models on both feature sets and the results from the vectorizer were almost 15% better at making predictions; consequently, we decided to train our final models using the vectorizer approach. A question that was out of scope for the current project, but would be of interest for future experiments, is how to generate a feature set that takes in contextual information of the sentences. This is discussed further in the conclusions section.

As our vectorizer relies heavily on the frequency of words, a word cloud visualization of positive reviews and negative reviews on our training set is depicted in Figure 1.

4.2 Model Results

Classifier	Validation Accuracy	Testing Accuracy
Naive Bayes (Multinomial TFIDF)	0.816	0.839
Naive Bayes (Multinomial Count)	0.803	0.827
Naive Bayes (Bernoulli)	0.760	0.825
Logistic Regression	0.884	0.885
Support Vector Machine	0.831	0.828
Decision Tree (Classic)	0.692	0.701
Decision Tree (Bagging)	0.767	0.764
Decision Tree (Boosting)	0.798	0.794
Random Forest (Regressor)	0.778	0.780
Random Forest (Classifier)	0.780	0.775

Table 1: Accuracy of Various Classifiers

As demonstrated in Table 1, logistic regression performs with the highest accuracy for both the validation and test set amongst all the classifiers. One possible reason for this may be that the feature set generated by our vectorizer returns a 100,000 dimensional object and logistic regression performs best in high-dimensional spaces, even when considering that most dimensions are actually just noise. Support vector machines and Naive Bayes also work decently well, matching our expectations in the abstract. On the other hand, classical decision trees underperformed our expectations, but again this might be explainable via the sparsity of the feature set.

We were able to improve the accuracy of decision trees by using a variety of ensemble methods including bagging and boosting. Additionally, we implemented a random forest classification model, another ensemble method in which the model weights the votes of multiple learners.

However, even after expanding the training data to include some of the validation data, the random forest model overfit slightly to the training data, and performed worse than logistic regression. Finally, we implemented a random forest regressor, which returned a value between 0 and 1, representing its confidence in the classification decision. We hypothesized that a random forest regressor would result in better accuracy as it would have more nuanced predictions regarding classification, thereby resulting in a less naive model. In order to turn the regressor's predictions into a classifier there are two methods that can be used: rounding and sampling. Sampling works by tossing a coin with the probability returned by the regressor. Under a uniform assumption, the rounding result would be the same as sampling, but since the underlying distribution was unknown, we ran on both algorithms. While it seems that sampling would be a more sophisticated approximation of the ground truth, rounding performed almost 10% better. Ultimately, while our random forest regressor achieved slightly better performance than the classifier, it still fell short of both our SVM and logistic regression models.

We also attempted two ensemble methods, random forest and boosting decision tree, where the features were the votes of our classifier. Since many of the classifiers were overfitting on the training set, we had to train the ensemble method on the validation data instead. Despite these additions, the results still showed a convergence to the accuracy of logistic regression.

5 Ensemble Methods

Given the set of classifiers, our next step was to determine if the addition of an ensemble method defined over the classifiers would result in a more accurate model. The most trivial method is to take the highest weighted classifier amongst those reporting (Boosting Decision Tree, Logistic Regressor, Naive Bayes, SVM). However, this method resulted again in a convergence of accuracy to the logistic regressor. In this method, we implicitly assign a uniform weight to every vote.

The next question was whether we could learn the optimal weights of each classifier. Our final approach was to include the votes of our classifiers along with the feature set generated by our vectorizer, and add an additional decision tree layer to make predictions off of this information. Ultimately, this method still fell short of our logistic regressor accuracy by a couple percentage points. We hypothesize the reason for this is that the logistic regressor just dominates all the other models, i.e. whichever inputs the logistic regressor gets wrong, the other classifiers also get wrong with high probability. As a result, the vote of the logistic regressor basically dictates the prediction of the ensemble method.

6 Conclusions and Further Discussion

In accordance with our experimentation agenda, we confirmed multiple machine learning models which performed with greater accuracy than the standard Naive Bayes approach. Logistic regression models reported the highest test accuracy amongst our experiments. Support vector machine models were shown to outperform all Naive Bayes models except Multinomial Naive Bayes with a TFIDF vectorizer. Surprisingly, even with the addition of various ensemble methods and data classifier techniques, logistic regression continually outperformed other classifiers.

Given that logistic regression proved to be the most fruitful approach amongst the tested classifiers, one technique that may be pursued with success in relation to sentiment classification is utilizing neural networks. Sigmoidal functions lend themselves well for binary classification due to the curvature of the function, and are frequently used as activation functions in neural networks.

With regards to future steps, it is also worthwhile to investigate the role that contextual information plays in binary sentiment classification. Consequently, one such neural network architecture that we would consider for future exploration is Bidirectional Encoder Representations from Transformers (BERT) [3]. BERT is a pre-trained model and has been shown to

boost state-of-the-art performance in many NLP settings. According to Xie et al.[4], BERT is able to achieve over 95% accuracy on the IMDB dataset.

Source Code

The code for our project is hosted on `github.com/brookekelseyryan/ColloquialTomatoes`.

References

[1] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[2] Saif M. Mohammad and Peter D. Turney. Crowdsourcing a word-emotion association lexicon. 29(3):436–465, 2013.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[4] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V. Le. Unsupervised data augmentation for consistency training, 2020.