COMP551-MiniProject2

John McGowan, Luis Pinto, Rebecca Salganik

February 2019

Abstract

In this project, movie reviews from IMDb are classified into reviews with positive sentiment and reviews with negative sentiment using various machine learning algorithms for classification. After pre-processing our data and transforming it using tokenization, we performed feature selection to find the best possible feature set. Results show that 91.2%, our highest accuracy on the test set, was achieved by linear support vector classifier and an ensemble of linear SVC and logistic regression.

1 Introduction

Given a data set containing 25000 IMBD reviews, various classifiers are used to predict the sentiment of the movie reviews. A Binary Naive Bayes classifier was implemented from scratch while built in Logistic Regression, Linear Support Vector Classification, and Random Forests were tested. In addition, combinations of these classifiers were tested. These models were trained on features extracted from the raw reviews. Count vectorization was used to make unigrams from specific lexicons and tf-idf vectorization to make up to tri-grams. Also *pattern.features* was used to assign polarity and subjectivity scores to each of the reviews. Different hyperparameters were tested for each built in classifier to determine the best classifier. Linear SVC, an ensemble of logistic regression and linear SVC, and an ensemble of linear regression, linear SVC, and random forest all achieved 90.7% accuracy using 4-fold validation set.

2 Related Work

Sentiment classification is by no means a new problem. In 2002 B. Pang and L. Lee explored the use of Naive Bayes and support vector machines in classifying the overall sentiment of documents. In *Thumbs up? Sentiment Classification using Machine Learning Techniques* they outlined some of the challenges in sentiment classification. P. Turney presented an unsupervised learning algorithm to classify reviews as recommended or not recommended which achieved 74% accuracy in *Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews.* Today, companies regularly use sentiment classification algorithms for feedback on their products. Although widespread, the use of sentiment classification in machine learning does not come without ethical conundrums. The law has not caught up with technology, so it is an open question as to weather an online review is the reviewers property or if by writing a review the reviewer consents to his or her review being used to train classifiers. As algorithms on social media outlets such as Facebook increasingly understand user sentiment and use that understanding to provide users with suggested content, these social media outlets have greater potential to influence public life, for example by polarizing electorates. As sentiment classifiers become more powerful, it is increasingly important that developers understand the ethical questions their algorithms raise.

3 Data Set & Setup

The data set provided to train our models is a list of 25000 IMBD reviews split in 12500 positive reviews and 12500 negative reviews. This split in the data implies that we will perform a binary classification rather than multi-class classification.
Before we could begin our feature selection process from the data we employed the methods below to make the data more likely to yield high accuracy predictions.

3.1 Text cleaning

First, we removed all punctuation and HTML tags from each review. This filtering out was achieved by using the *re.compile* package. Furthermore, we put each word into lowercase form in order to normalize them. We also explored other forms of text transformations such as stemming and lemmatization but ultimately chose to remove as they did not improve performance.

3.2 Pre-processing

After the text cleaning, we used tokenization to separate the text into a set of words delimited by white spaces so they could be represented as vectors as explained in the Feature section below. The maximum amount of entities per token we used was three (tri-grams).

4 Proposed Approach

In this project, we experimented with four standard algorithms: Naive Bayes classification, logistic regression, linear support vector machine and random forest. Also, a combination of the mentioned classifiers are tested to improve our ability to generalize over a single estimator. The chosen metric used to measure performance is accuracy. In order to evaluate them, we must reserve a portion of the training data for the validation set. This was done using a held-out validation set for the Naive Bayes approach and using a 4-fold cross validation for the rest of the classifiers.

4.1 Classifiers

4.1.1 Binary Naive Bayes

Binary Naive Bayes is a linear classifier based on Bayes Rule (equation 1) which defines a boundary based on training data and classifies new inputs based on where they fall with respect to the boundary. The boundary is defined by equation 2 and 3 where the x_j are binary features. In this paper, the features will be weather or not a word appeared in a movie review. If $\delta(x) > 0$ the review is classified as a positive review, while if $\delta(x) < 0$ the review is classified as a negative review.

4.1.2 Logistic regression

The logistic regression model is a binary classifier which uses a logistic function to determine the probability of a value belonging to a class.

4.1.3 Linear support vector classifier

Linear support vectors classifier (SVC for short) are a form of binary classifier which uses value mapping in a hyperplane to determine a decision boundary which divides the objects into two classes. The motivation for this model comes from the idea of a decision boundary. The intuition behind the SV Classifier is that looping through misclassified data points and correcting their weights can lead to a simply calculated high prediction accuracy.

In our experimentation we found the most influential parameters for both logistic regression and SVC to be the penalty and C values. Both of these values refer to different aspects of regularization which were applied the classifier. Regularization can be seen as a method for counteracting overfitting because its main function is to penalize models with a large number of features. We found that our best accuracy was achieved through the implementation of L2 regularization.

4.1.4 Random Forest

This classifier is an ensemble of decision trees which are trained from randomly selected subsets of the data set. Then, the random forest averages these trees to create a low variance model. We chose to change the default number of estimators (i.e. decision trees) to increase predictive power.

4.1.5 Ensemble Classifiers

Ensemble classifiers are meta-algorithms that combine different machine learning techniques into one predictive model. In order to balance out the individual weaknesses of the classifiers mentioned above, combinations of them are used such that the predicted class label for a particular sample is the class label that represents the majority of the class labels predicted by each individual classifier. In case of a tie, the class label will be selected based on the ascending sort order. Two ensembles were used: a combination of logistic regression and linear SVC; and a combination of logistic regression, linear SVC and random forest.

4.2 Features

A vector representation of a text is needed for the classifier to calculate probabilities or decision boundaries. After pre-processing the text data, we can proceed to make such representation of the text by count and by tf-idf. Note that all features are l2 normalized in the feature pipeline.

4.2.1 Term frequency

A count vectorizer creates a count of each individual token's mentions (in this case only uni-grams). For all classifiers, all non-zero counts are set to 1 so the total count becomes binary, i.e. either seen or not seen. Two lexicons (i.e. which words to include vs. ignore) were used: the Bing-Liu's opinion lexicon and a list of words specific for movie reviews from the *nltk.corpus*. Furthermore, the first lexicon is split in two: positive and negative words.
Tf-idf is the product of two statistics, token frequency and inverse document frequency. This feature represents the number of times a given token appears in a review relative to the total number of reviews containing that token.

4.2.2 Adjectives

Using the package *pattern.sentiment* we were able to use the built in lexicon of adjectives that occur frequently in product reviews to assign polarity score and a subjectivity score to each review. It ended up not improving the accuracy of any of the classifiers.

4.2.3 Number of words

The total number of words in each review (length of the review) was an attempt of a feature because we estimated that feature length was related to depth of emotion. However, just like the adjectives feature, it did not help improving the accuracy.

4.3 Feature selection

Each feature is normalized to l2 norm equal to one for better predictions. The Binary Naive Bayes model was trained using four combinations of the lexicons: only positive words from Bing-Liu's opinion lexicon, only negative words from Bing-Liu's opinion lexicon, both positive and negative words, and both positive and negative words along with a list of words from movie reviews. For the rest of the classifiers, the score from the chi squared test was used to select the features with the highest values. The package *SelectKBest* from *sklearn* was applied to improved accuracy and helped reduce the likeliness of the model to over fit.

5 Results

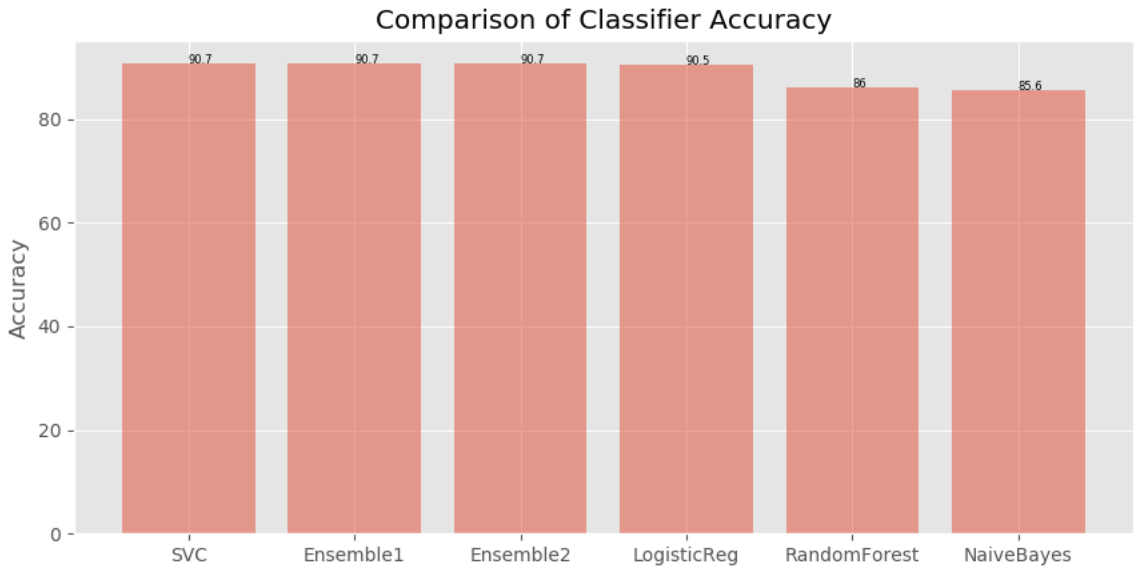

Figure 1: Overview of Classifier Prediction Accuracy Values

The highest accuracy on the validation set was 90.7%, achieved by three of our classifiers. The first one is the linear SVC with an l2 norm used for penalization, 4000 maximum number of iterations and C = 100, trained on the top 1 million tf-idf uni-grams, bi-grams and tri-grams given by the chi squared test. Also, the ensemble of logistic regression and linear SVC got the same accuracy. The hyperparameters of each classifier in the ensemble were the same as the ones used for the best linear SVC mentioned above. The features this ensemble used were the top 53000 features from uni-grams, bi-grams and tri-grams vectorized by tf-idf and uni-grams vectorized by count using the complete Bing Liu's lexicon as its vocabulary. Lastly, the third model to achieve this accuracy is the ensemble of logistic regression, linear SVC and random forest. Just like the previous model, the hyperparameters used on the first two classifiers are the same. For the random forest classifier, the number of estimators was increased from its default value to 150. Also, it used the top 530000 features from the same combination of features used in the first ensemble.

Logistic regression model achieved 90.5% accuracy using the same hyperparmeters as the best linear SVC. However, it was trained using the top 70000 features from uni-grams and bi-grams vectorized by tf-idf.

Random Forest achieved 86% accuracy with 500 estimators, slightly outperforming our best Binary Naive Bayes model. Also, the default criterion 'gini' (referring to the function used to evaluate the

quality of split) performed significantly better than that of its counterpart, "entropy", which measures information gain.
The Naive Bayes model performed best with the full Bing-Liu opinion lexicon in combination with the list of words from movie reviews giving an accuracy of 83.7%. A plot of the accuracy of each model is shown in figure 2. In this case, 21250 reviews were used to train and 3750 reviews (15% of the data set) were held out to test.

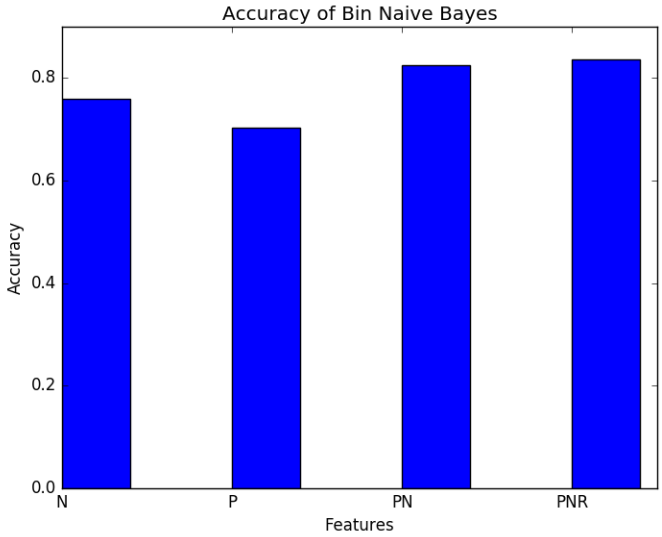

Figure 2: Accuracy of Binary Naive Bayes trained on different list of words. 'N' stands for negative words only, 'P' for positive words only, 'PN' for both positive and negative words, and 'PNR' for positive words, negative words, and words from reviews. From left to right, the percentages are 75.9%, 70.34%, 82.4% and 83.7%

6 Discussion & Conclusion

Three classifiers achieved 90.7% accuracy on a 4-fold validation set: linear SVC, an ensemble of of logistic regression and linear SVC, and an ensemble of linear regression, linear SVC, and random forest. Note on the kaggle test set, linear SVC and the ensemble of linear SVC and logistic regression scored 91.266%, outperforming the ensemble of lnear SVC, logistic regression and random forest.
It was interesting to note the variability among the classifiers when using untrained hyper parameters. For example, the random forest classifier (which eventually achieved a prediction accuracy of 86% was initially performing at approximately 49% when it was run with criterion = 'entropy' setting. Although our testing was extensive, it was not exhaustive and such findings raise the interesting possibility for further methodical optimization of parameters.
Furthermore, as noted in the related works, the predictive qualities of lexicons plays a very significant role on feature selection and through it, classifier performance. When reading over the Bing Liu's opinion lexicon we found that many of the selected words were not those a human might have chosen as strongly predicting sentiment. There is much room for experimentation in the creation of lexicons of words that predict positive or negative sentiment.

References

[1] Bo Pang and Lillian Lee. 2002. Thumbs up? Sentiment Classification using Machine Learning Techniques *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*

[2] Peter D. Turney. 2002. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. *Proceedings on the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*

Appendix A Statement of Contributions

Naive Bayes Classifier: John McGowan
Alternate Classifier Testing: Luis, Pinto, John McGowan, Rebecca Salganik
Write Up: Rebecca Salganik, John McGowan, Luis Pinto

Appendix B Equations

$$P(y = 1|x) = \frac{P(x|y = 1)P(y = 1)}{P(x)} \tag{1}$$

$$\delta(x) = log\frac{P(y = 1|x)}{P(y = 0|x)} = log\frac{P(y = 1)}{P(y = 0)} + \sum_{j=1}^{m} w_{j,0} + \sum_{j=1}^{m}(w_{j,1} - w_{j,0})x_j \tag{2}$$

$$w_{j,n} = log\frac{P(x_j = n|y = 1)}{P(x_j = n|y = 0)}. \tag{3}$$