Ry Rutherford
260784558

Sentiment Analysis In Movie Reviews

Problem Setup

The goal of this project was to determine which machine learning classification model and preprocessing steps were most accurate at predicting positive or negative sentiment in sentences extracted from movie reviews. I used a dataset of 10662 sentences extracted from Rotten Tomatoes movie reviews of which half were tagged positive and half were tagged negative.

Experimental Procedure

I used five models in the experiment from the scikit-learn library: Multinomial Naive Bayes, Gaussian Naive Bayes, SVM with a linear kernel, Logistic Regression, and a dummy model which guessed positive or negative with equal probability.

Before training the models I had to preprocess the data. Feature extraction was done using scikit-learn's CountVectorizer and TfIdfTransformer classes which resulted in a Tf-idf term weighted feature vector on unigram counts. I experimented with lemmatization and Porter stemming, removing words that appeared strictly fewer than 5, 7 and 10 times and using stop words. I also experimented with doing no preprocessing.

I used three different training and test set sizes (60/40, 70/30, and 80/20 splits) in order to find out which sizes led to the best performance. I used scikit-learn's GridSearchCV class to discover the optimal hyperparameters for each model by performing 5-fold cross validation across a specified range of hyperparameters on the training set. I trained each classifier on all possible preprocessing and training size combinations leading to 72 unique models per classifier.

In order to determine which of the 360 models performed best I used scikit-learn's classification_report function to calculate each model's overall accuracy along with the precision, recall and f1-score. I compared the output of each classification report to identify the top performing models for each type of classifier.

Parameter Settings

In the table below, I show which hyperparameters were tested in GridSearchCV for each non-dummy classifier. Bolded values represent the hyperparameters that led to the best accuracy.

	MNB	GNB	LR	Linear SVM
Hyperparams	fit_prior=**True**, False alpha=0.001, 0.1, **1.0**	var_smoothing= **0.2**, 0.5, 0.7	C=0.5, 2.0, **3.5** tol=**0.005**, 0.0001 max_iter=**1000** dual=**False**	C=**0.4**, 0.55, 0.7 max_iter=75, **100** class_weight= **balanced**, None dual=**False**

Results and Conclusions

Multinomial Naive Bayes was my best performing model with an accuracy of 77.84%, (see the confusion matrix at the end). This model was trained on 60% of the corpus with stopwords

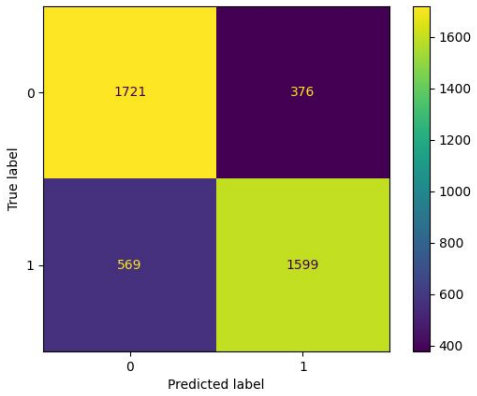
removed, no text normalization and no words removed based on their frequency. The model used the default alpha hyperparameter (1.0) and had fit_prior=True. Gaussian Naive Bayes, Linear SVM and Logistic Regression achieved the next best accuracy with 76.99%, 76.60% and 76.55% respectively. As expected the dummy model was the least accurate with 52.00% accuracy.

The top performing models for each classifier were trained on data that did not undergo text normalization. This may be because different forms of words can be judged to carry different sentiment and when they are stemmed or lemmatized to one word they are all interpreted the same way. Hence the model may be less accurate when it sees a stemmed or lemmatized word.

Further, the removal of stop words did not seem to have much of an effect on the performance of a model. This may be explained by my use of a Tf-idf feature vector which lowers the weight of high frequency words in the data set while stop word removal performs a similar function by removing features corresponding to the most common words in a language. It is likely there was a high overlap between the most common words in the dataset and nltk's stop words list; therefore, removing stop words would not result in a significant impact on performance.

Removing infrequently occurring words generally seemed to have a negative effect on model performance, especially when removing words that appeared fewer than 10 or 7 times. This is probably because there were some infrequently used words that were only used in negative or positive contexts in our corpus and hence provided the model with useful information.

Given that both Naive Bayes models were the best performing models it can be speculated that the features were somewhat close to being conditionally independent and thereby followed the Naive Bayes bias. When tuning the hyperparameters for Logistic Regression and Linear SVM there was a noticeable difference in performance based on the level of regularization used (e.g. 2-3% accuracy improvement when not using the default values). Therefore, it is plausible that Logistic Regression and Linear SVM could achieve similar or better performance than the Naive Bayes models if more granular values for regularization were tested in GridSearchCV.



Confusion Matrix for Best Performing Model
Label 0 = Negative Sentiment, Label 1 = Positive Sentiment