Samantha Riedener

IST 736 HW 7

Introduction

Multinomial Naïve Bayes and support vector machines are two of the most common tools to be used for making predictive models. While their underlying mechanics are vastly different, they’re still both incredibly useful when trying to make predictions and are worth looking at side by side to better understand how effective each of them are. Since the same success measures can be used with each of the models made from these algorithms, it’s a simple task to be able to compare how they perform.

For the purposes of this task several Naïve Bayes and support vector machine models will be built with the same parameters in order to measure how they perform against each other. In successive models, one major change will be made to isolate how that change effects the overall output of the predictions. A final support vector machine will also be made without a comparable Naïve Bayes model to examine how well it can perform under a different method of success evaluation.

Analysis and Methods

The data set that is being analyzed is a collection of sentences and phrases that have been assigned a sentiment value from 0-4, with 0 being most negative and 4 being most positive. The data is read in and split into its phrases and its sentiment values. This is then split further into a training set and a testing set with a 60/40 split. The distribution of the sentiment scores is then checked to get a baseline for success of the models, expressed as both raw counts and as a percentage of the data set:





Based on this output, the neutral sentiment category makes up the majority of the data set at nearly 51% of the points being in that category. Therefore, if the predictions were to just do a majority vote, they would be correct about 51% of the time. For their predictions to be considered successful, they need to achieve a higher than 51% accuracy.

Two vectorization models were built to be fed into both algorithms: one counted only unigram tokens and the other counted unigram and bigram tokens. English stopwords were also set to be removed and only tokens that appeared in at least 5 of the documents would be counted. These second two options were chosen given the large size of the data set; the algorithms were more likely to perform better by trimming down some of the most and least common words so that meaningful predictions could be made. The choice to explore both unigrams and bigrams was done to see if tokens consisting of two words could be useful to the algorithms. The dictionary size of both these vectorizers was checked and found to be:

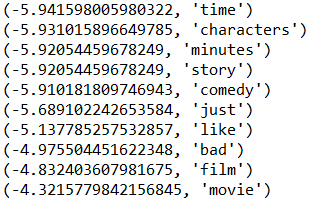
Unigrams only – 11,967

Unigrams and bigrams – 34,579

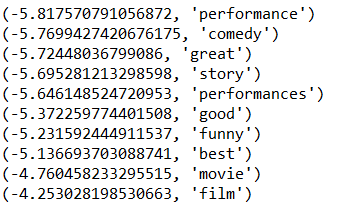
These results are in line with what would be expected as the second vectorizer would logically have a larger dictionary size due to it counting one word and two word tokens (for example, “big”, “enough”, and “big enough” would be counted as three tokens in the second vectorizer, but only as two in the first).

The vectorized data set was then run through the algorithms to train it for further predictions. For the first vectorizer only, the top ten words in both the most positive and most negative categories were explored for both Naïve Bayes and SVM to see what they had learned. Only the first vectorizer was explored because the second one contains bigrams and those cannot be split into individual words, which was the purpose of this exploration. The results were as follows:

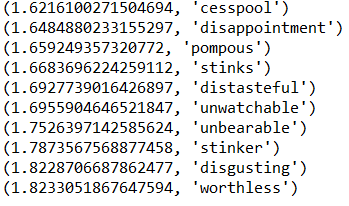
Naïve Bayes, Very Negative



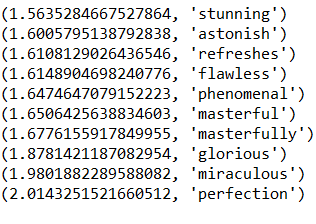
Naïve Bayes, Very Positive



SVM, Very Negative



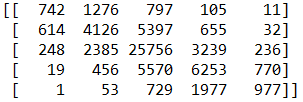
SVM, Very Positive

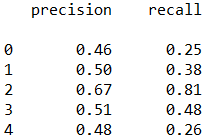


Next, the quantitative performance measures were calculated for each of the four algorithms. This was done with a success rate, a confusion matrix, and with precision and recall scores. The results for each of them were:

Naïve Bayes, Unigram

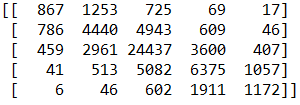
Success rate: 60.6%

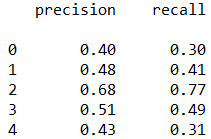




Naïve Bayes, Unigram and Bigram

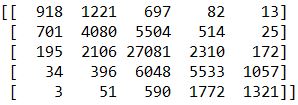
Success rate: 59.7%

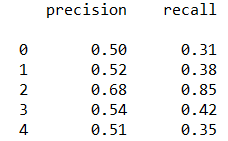




SVM, Unigram

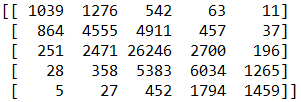
Success rate: 62.4%

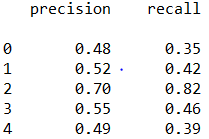




SVM, Unigram and Bigram

Success rate: 63.0%





It should be noted that the results above are the best results that were achieved after changing various constraints in the vectorizer.

For the final model, an SVM model was made using the entire training set. Since this left no test data to measure accuracy, this meant that a k-folds measurement had to be taken to check the accuracy. A 5-fold cross validation was chosen as a balance between time and accuracy. The results shown below reflect the highest possible average given the alterable parameters in the vectorizer and the algorithm:



Average: 56.1%

Results

Based on the initial metric of greater than 51% accuracy to be considered successful, all the algorithms did perform successfully. The algorithm that performed the best was the SVM algorithm that had looked at the data set that had been vectorized with both unigrams and bigrams. Not only did it achieve the highest accuracy, but it also had the highest precision and recall in the most categories: 1, 2, and 3 for precision and 0, 1, and 4 for recall. The SVM algorithm that only looked at unigrams had the highest precision and recall where the other SVM algorithm did not, except for the recall in category 3, which was highest in the Naïve Bayes algorithm that looked at both unigram and bigrams. It is still sufficient to say that the SVM algorithms outperformed the Naïve Bayes algorithms.

Looking at the precision and recall values across each of the algorithms, there’s a pattern in what they managed to do best. Category 2 consistently had the highest values while categories 0 and 4 had the lowest. This makes sense as in the training set, category 2 had far more data points than any other category, making it the easiest for the algorithms to learn.

The final SVM algorithm performed less well when using the full data set and k-fold evaluation than the split training and test data. While the vectorizer settings were kept the same, after some experimentation, it was found that reducing the error cost from 1 to 0.5 produced slightly better overall results. It is still unclear why the k-folds method produced a lower success rate than the 60/40 training split because 5 folds would leave a larger training set to learn from.

Conclusion

Between the two algorithms, SVMs outperformed Naïve Bayes. While the margin was not large, it was still noticeable and may lead to SVM being chosen more often in future text analysis. While SVM did take more time to build, the trade off was still worth it for this data set to invest that extra time to get the better results. Different tasks may still prefer Naïve Bayes for its much faster processing time, even if it doesn’t perform quite as accurately as SVMs can.

This exercise looked only at linear SVM models, which may have been limiting. It is generally true that linear SVMs perform the best with text analysis, but it could be an avenue worth exploring with this data set in the future. Furthermore, there are many data processing steps that could be applied to this data set to try and achieve better results. Things like stemming or lemmatizing could be applied and given the size of the data set these techniques should not be too limiting to what the algorithms could find.