**Introduction**

Sentiment analysis is a useful natural language processing technique where a model is created using text data and can classify the sentiment of new phrases. The dataset used for this project comes from a Kaggle competition and is movie reviews. Each review has been split into subsections and a sentiment has been given to the original review and its corresponding subsections. For this project only the original review (no subsections) and accompanying sentiment was used. The goal of this project is to create a model that can take in new movie reviews and classify it according to the sentiments from the training data. The possible sentiments range from 0 : very bad, 1 : somewhat bad, 2 : neutral, 3 : somewhat good, and 4 : very good. The cleaned dataset consists of 8,530 movie reviews and its accompanying sentiment. The spread of sentiments across these reviews is shown below in figure 1.

A picture containing text, screenshot, diagram, font

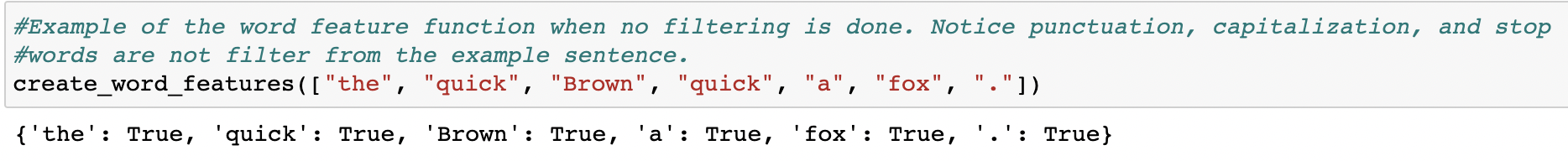
Description automatically generated

*Figure 1.* Sentiment spread in movie reviews dataset

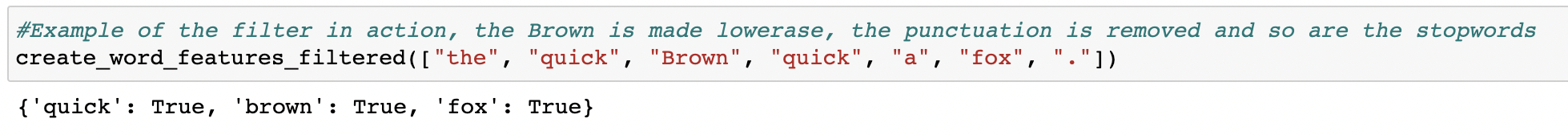
**Text Processing**

This data originally had 156,060 rows, with 4 columns: PhraseId, SentenceId, Review, Sentiment. The phrase id was the unique key, the sentenceId corresponded to each review, again each review had the original review and subsets of the review, and all phrases had an accompanying sentiment. A phrase length column was created and the longest phrase for each sentenceId was compiled into a dataset that would only contain the full reviews. The final used dataset was 8530 rows long.

The first step in processing this text involves creating word features. A word feature will read the words from a text sentence, tokenize it, filter it however needed, and put the tokens into a dictionary with an accompanying TRUE. In this way the keys to the dictionary would be the tokens and the values would be true. This dictionary was the first part of a tuple, and the second part was a value 0:4 which was the sentiment of the review. This was found to be a useful way to tokenize words as filters can be added/subtracted from the reviews. To better visualize this a demonstration is shown below. The first bit of code doesn’t filter anything, the second bit filters stop words, punctuation, and capitalization.



*Figure 2.* Tokenizing without filtering



*Figure 3.* Tokenizing with filtering

It was found from experimentation that filtering could remove useful information, so the base model that will be the default did not use filtering. This will be discussed more in depth in the experiments section.

**Feature Engineering**

A multitude of different features were used in creation of a classification model. The most basic of these will be discussed here, and if more information on the features used in the experiment section are of interest, the whole annotated python file will be attached to this report. The entirety of this most basic feature can be seen below, the intention is to take the 2000 most common words from the movie reviews, create a featureset containing them, then read each movie review and tag the particular words that were in that review and the accompanying sentiment. Doing this would allow a naïve bayes classifier to be applied and classification could be completed this way. Finally, a 5-fold cross validation was completed, and an accompanying confusion matrix can be seen.

A screenshot of a computer code

Description automatically generated with medium confidence

*Figure 4.* Default base feature set code example

To begin a function called document\_features was created that would take the movie reviews and a featureset input and for every word in the review that matched a word in the featureset it would mark the word as True. Next the featureset was created by taking the 2000 most common words in the reviews. A cross validation feature was created that would take the given feature set and the number of folds, and test and train the data using naïve bayes and output the accuracy, this would be repeated for every fold. Finally, the predicted sentiments were compared to the actual sentiments using a confusion matrix, which is the eval\_measures function. So in summary, the movie review would be fed into the code, every word in that movie review that matched a word in the most frequent words would be marked, the data would be split into training and testing data, naïve bayes would train on the training data and the testing data would be fed in for the Naïve Bayes prediction, the prediction would be compared to the actual sentiment, and a confusion matrix would be created to see where there are issues.

It should be noted that this is the most basic, default bag of words approaches, and furthermore complex approaches will be addressed in the experiments section. The code and summary were included for clarity and again, the accompanying code will be attached to this report.

Finally, cross validation was chosen in conjunction with confusion matrices for several reasons. To help discuss this, the 5-fold cross validation and resulting confusion matrix and output statistics are shown below for discussion.

A picture containing text, font, number, screenshot

Description automatically generated

*Figure 5.* Output statistics and confusion matrix

Cross validation helps alleviate anomaly model responses where the sampling happened to be very favorable or unfavorable. Essentially, the data is sampled into testing and training sets and an accuracy using Naïve Bayes is done multiple times (for this project 5-fold cross validation was used). Doing so tests the efficacy of this data 5 times and an average accuracy can be calculated. This is more reliable than a single accuracy reading. Additionally, using confusion matrices allows for further analysis like accuracy, recall, precision, and F1. Accuracy is the number of correct classifications divided by the number of testing data points. If the model guessed the correct sentiment 8 times and 10 movie reviews were tested, it would have an accuracy of 80%. Recall, or sensitivity, is the number of times the model classified the review correctly divided by the number of correctly classified reviews plus the number of reviews with that sentiment that were incorrectly classified. For example, looking at the confusion matrix for reviews with a sentiment of 3. The precision would be the number of times the model correctly classified a sentiment of 3 was a sentiment of 3 (12.2%). This would then be divided by that same 12.2%+all the times it truly was a 3 but the model guess something different. For this example, it would be the top row (5.6%, 4.8%, 5.2%, and 1.0%). This would give which can be seen for the precision of sentiment 3 just above the confusion matrix. The recall is very similar however it’s the number of correctly classified sentiments over the correctly classified sentiment plus the number of times a 3 sentiment was misclassified giving which again is seen above the confusion matrix. Finally, the F1 statistic is calculated by . The F1 statistic is the easiest way to quantify the results that encompasses both precision and recall, an F1 closer to 1 is better.

**Experiments**

Experiment 1

With the basic model understood, different feature set experiments were performed and compared to the default baseline. The first experiment involved filtering the data as shown above in Figures 2 and 3. The filtering removed punctuation, stop words, and capitalization. This experiment did not use the bag of words approach as before, instead it simply trained on which words were present in the movie review, and no bag of words was used. Using filtering decreased the accuracy of the 38.43% to 35.56%. It should be noted that cross validation was not used in this experiment as it was a quick display to see how filtering affected the data. Every time this runs, it will produce slightly different results. However, removing stop words and punctuation was not used as some punctuation may provide insight for the model. A nice example of this is shown below, the most important features of the training data is shown, and a question mark can be seen as the 3rd most important feature. People used question marks when they had a neutral opinion, and for this reason the punctuation was kept.

A picture containing text, receipt, font, white

Description automatically generated

*Figure 6.* Most informative Features Experiment 1

Experiment 2

The second experiment is using the code from figure 4, however instead of the 2000 most common words making the feature set, the most common 4000 words were used. Using more words or datapoints was hypothesized to increase the amount of data that the model could train on and hopefully increase accuracy. The 5-fold cross validation, output statistics, and confusion matrix is seen below in figure 7. The accuracy is nearly identical to the default 2000 frequency words, from 37.61% to 37.66%. The accuracy is essentially identical; however insights can be made. The F1 statistic are worst for sentiment 0 and sentiment 2. Note that sentiment 0 is very negative and sentiment 2 is neutral. It’s a common theme that neutral data is very difficult to classify in this dataset. Additionally, looking at the confusion matrix, there is often issues differentiation sentiment 1 from 0 and 3 from 4. This is because sentiment 0 and 1 are both negative but to different degrees and sentiment 3 and 4 are both positive to different degrees. There is much less of an issue resolving a sentiment 0 and a sentiment 4 for example.

A picture containing text, font, number

Description automatically generated

*Figure 7.* Experiment 2 output

Experiment 3

The third experiment involved analyzing the data with the understanding that negation words are present. Negation words are things like “not” or “hardly”, that can transform a positive movie review into a negative one. For example, “That movie was worth watching” and “That movie was hardly worth watching” are nearly identical sentences however, the negation word “hardly” completely changes the sentiment. If a negation word + feature word appeared in the dataset the negation phrase was added to the featureset. The featureset would included words such as “good”, “bad”, but also phrases such as “not good”, “not bad”. In this way, the negation of certain phrases was accounted for. The 5-fold cross validation, output statistics, and confusion matrix is seen below in figure 8. Again the issue with this dataset is the sentiment 2. A stagnant accuracy is again observed with the accuracy of 35.94%. Again, the same issues of mixing sentiments 1 and 0 as well as 3 and 4 was observed again.

A picture containing text, font, screenshot, number

Description automatically generated

*Figure 8.* Experiment 3 output

Experiment 4

The fourth experiment used the same bag of words approach. However, instead of just using the most frequently occurring words, the most frequently occurring bigrams was also included. This pushed the feature set from the original 2000 words to 10,000 features, as the most common bigrams and most common words were used. The bigrams were found using chi-square and the training and testing data was created in the same way. The 5-fold cross validation, output statistics, and confusion matrix is seen below in figure 9. Using the bigrams did increase the accuracy slightly up to 37.7% which was the best accuracy achieved for the techniques learned in lab. Again, the main issues were identical to those observed for other experiments. It was very difficult for the model to differentiate somewhat good and good as well as somewhat bad and bad. The neutral sentiment again couldn’t be classified, often being guessed as either sentiment 1 or 3. It should be noted that the highest F1 score across every experiment was a movie sentiment of 1. The model was able to best predict if a movie review was slightly negative.

A picture containing text, font, screenshot, number

Description automatically generated

*Figure 9.* Experiment 4 output

More Advanced Task

After reviewing what the winners of this sentiment competition were using, it was clear that regular sentiment analysis using word frequencies and particularly naïve bayes wouldn’t be powerful enough to get accuracies that exceeded 40%. The best accuracies were achieved with neural network models like BERT. To try to increase the accuracy of my own models, SKLearn was used, and a support vector machine was created. The same bag of words approach was used as before, however, this time the training data was used on a support vector machine and the testing data was ran through the SVM model. The results of this can be seen below in figure 10. This model was the best, with an accuracy of 40.65%. The SVM was able to classify the sentiment 1 and 3 most effectively. Interestingly it also struggled with sentiment 0 and 2 the most. This may be an indication that the data itself is difficult to understand simply with featuresets.

A picture containing text, font, screenshot, number

Description automatically generatedA picture containing screenshot, rectangle, square, colorfulness

Description automatically generated

*Figure 10.* SVM output

**Conclusion**

In conclusion many things were achieved/learned throughout this process. The no information rate of this data was 27.25%. 27.25% of the data was sentiment 3, so if the model simply guessed 3 for all testing reviews it would be right around 27.25% of the time. Models using all experimental measures were created that exceeded the 27.25%. No accuracy over 40% was achieved using the more simplistic approaches discussed in lab. The best model was created using a support vector machine and yielded an accuracy over 40%. The largest struggle with classifying this data was the neutral sentiments. Classifying a review as neutral indicates that there aren’t any strong classifying words in the sentence itself. For it to be neutral, it must lack highly polarizing words. The best insight found about neutral data from this dataset was that people often asked questions or used the “?” when they had a neutral opinion. The neutral reviews hindered the accuracy to below 40%. To further push that sentiment accuracy, different language models like BERT would need to be employed. Finally, it should be noted that this data was labeled by humans, and would of course introduce bias and potential error, so no model will ever be perfect.