Barbara Schweitzer

Natural Language Processing Final Project Report

With the shift towards online shopping in the last 20 years, and an exponential increase in the last year and a half, huge amounts of data have been produced from these transactions. One such source of data is product reviews, which can be found on nearly every product that is sold online. The data set chosen for this analysis is a dataset consisting of Amazon Musical Instrument reviews.

The first step in the analysis of the musical instrument review data was to process the data. First, the data was loaded into Python and the unnecessary columns were removed. In this case, these columns were reviewerID, asin, reviewerName, helpful, summary, unixReviewTime, and reviewTime. Due to the nature of this text analysis, these fields were unnecessary. After this processing, the reviewText and overall columns were remaining. The reviewText column contains the words from the review left on an item, and the overall column was the numeric value of the number of stars left with the review. To simplify the analysis later on, a third column, titled ‘target’, was added. The values in this column correspond to the star rating from each review. If the ‘overall’ value was greater than or equal to 4, the entry would receive a 1 as the target entry. For star ratings of 3 or smaller, the target variable would have a value of 0. This simplified the rating scale, making the reviews have a ’positive’ or ‘negative’ rating. To complete the data processing, the reviewText column was changed to the string data type.

The next step of the analysis was to create a bag-of-words, made by finding the most frequently used words in the reviews. For this step, I chose to filter the review text, as I felt that it would produce a more meaningful result. For this filtering, I created an empty array for the text. I then removed the non-alphabetic characters, including punctuation and numbers. Next, I converted all letters to lower cases, and removed stop words. After lemmatizing the text, it was appended into the empty array. Next, it was necessary to find the word frequencies. To do this, the corpus was tokenized, and an iterative function was created to count the occurrences of each word. From this frequency distribution, the 1,500 most common tokens were chosen. The most frequently used word in the corpus was ‘guitar’, which was found 6,847 times, followed by ‘one’, occurring 4,946 times, and ‘string’ 4,724 times.

Using the previously constructed bag-of-words, a word cloud was created from the most frequently occurring words.

A picture containing text, newspaper

Description automatically generated

As previously stated, ‘guitar’, ‘one’, and ‘string’ were the most frequently used words. With the word cloud, we can clearly see these three words stand out as the largest, along with ‘sound’, ‘great’, and ‘like’.

The next portion of the analysis is the Naïve Bayes models, created using NLTK. To first begin building these models, the filtered review text was added to the original data frame, in a column titled ‘cleanText’. Again, this filtering consisted of changing all cases to lower cases, removing non-alphabetic characters, removing stop words, and lemmatizing the text. For the first model, two new data frames were created. The first contained the positive reviews, or the reviews with a star rating greater than 3. The second contained the negative reviews, or the reviews with star ratings less than 4. The cleaned text in each new data frame was joined into a list of strings, where it was then tokenized using the White Space Tokenizer. The respective lists of tokens were then concatenated with their positive or negative rating, resulting in a list of words with the overall rating of the review it was from. After this step, the positive and negative word lists were recombined and shuffled. Below is a preview of the final word list.

Text

Description automatically generatedThe training and testing sets were then created using a 70/30 split. Finally, the Naïve Bayes model was created and tested on the testing data set. This model resulted in an accuracy of 85.50%. To hopefully improve the validity of the model, 5-fold and 10-fold cross validation was done. For the 5-fold cross validation, the resulting mean accuracy was 85.65%, which was marginally better than the standard model. For the 10-fold cross validation, the mean accuracy was 85.67%, which was again marginally higher than the first model. To get a better idea of the validity of the model, the precision, recall, and F-measure

scores were calculated. The following table shows the scores for both the negative and positive reviews.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F-Measure** |
| **Positive** | 0.016 | 0.293 | 0.030 |
| **Negative** | 0.994 | 0.859 | 0.922 |

For the negative reviews, the three scores were very high. For the positive reviews, the scores were quite low, likely due to the smaller quantity of positive reviews in the data.

For a second model, the same Naïve Bayes model was created, however this model used the original review data, with no filtering. Following the same steps of creating the positive and negative data sets, labeling, and then recombining them, below is a preview of the words used.

Text, letter

Description automatically generatedClearly, the words used in this model were not as meaningful as the words used in the first model. Using this unfiltered list, the training and testing sets were created, again using a 70/30 split. The model created had an 85.35% accuracy. Again, cross validation was performed. The 5-fold cross validation resulted in a mean accuracy of 85.42%, and the 10-fold cross validation resulted in a mean accuracy of 85.43%. though lower than the accuracies of the last model, there was not a large discrepancy between the scores. Next, the precision, recall, and F-measure scores were computed, and are shown below.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F-Measure** |
| **Positive** | 0.028 | 0.294 | 0.051 |
| **Negative** | 0.989 | 0.860 | 0.920 |

These scores were slightly higher for the positive reviews, and approximately the same for the negative reviews, as compared to the first model.

For the final NLTK Naïve Bayes, a third version of the review text was used. In this case, the data was filtered less than the first model, by changing all cases to lower cases, non-alphabetical characters were removed, and the text was lemmatized. This review text was added to the original data frame in a column named ‘cleanishText’. Following the same steps of creating the positive and negative data sets, labeling, and then recombining them, below is a preview of the words used.

Text

Description automatically generatedWith the stop words remaining in the review text, this list of words was slightly less meaningful than the first models, but more so than the previous model with no filtering. Using the same 70/30 training and testing split, the Naïve Bayes model was created. This model resulted in an accuracy of 85.33%. After both 5-fold and 10-fold cross validation, the mean accuracies were 85.46% and 85.46%. again, these accuracy values were slightly lower than the first model, though they were comparable. Again, the precision, recall, and F-Measure scores were calculated for the model. These are shown in the table below.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Precision** | **Recall** | **F-Measure** |
| **Positive** | 0.028 | 0.475 | 0.054 |
| **Negative** | 0.995 | 0.861 | 0.923 |

The three scores above, for both positive and negative reviews, were higher than the previous two models, suggesting that this may have been the most effective model for the given data.

As a summarization of these three models, below is a table representing their scores.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Accuracy** | **5-Fold CV Accuracy** | **10-Fold CV Accuracy** | **Precision** | | **Recall** | | **F-Measure** | |
|  | | | | **Pos** | **Neg** | **Pos** | **Neg** | **Pos** | **Neg** |
| **Model 1** | 85.50 | 85.65 | 85.67 | 0.016 | 0.994 | 0.293 | 0.859 | 0.030 | 0.922 |
| **Model 2** | 85.35 | 85.42 | 85.43 | 0.028 | 0.989 | 0.294 | 0.860 | 0.051 | 0.920 |
| **Model 3** | 85.33 | 85.46 | 85.46 | 0.028 | 0.995 | 0.475 | 0.861 | 0.054 | 0.923 |

Based on the score comparison for the three models, there does not appear to be a clear winner. For each model, the accuracy remained in the 85% range, only slightly increasing with cross validation. The precision, recall, and F-Measure scores for all three models were quite poor for the positive reviews, likely caused by the lack of positive reviews in the data.

For a more complex, advanced model, I chose to create two Multinomial Naïve Bayes models using SKLearn, with a Term-Frequency Times Inverse Document-Frequency (TF-IDF) transformer. The first model created utilized the cleaned review text data; the same data used in the first model. The first step consisted of vectorizing the data using the SKLearn CountVectorizer function. Applying the SKLearn TfidfTransformer, this data was then split into training and testing data, utilizing the same 70/30 split as the previous models. This model resulted in an accuracy of 88.67%. Again, utilizing cross validation, with both 10-folds and 20-folds, the resulting mean accuracies were 87.58% and 87.57%. These accuracies were better than the previous three models run. However, when computing the precision, recall, and F-Measure scores, the values were 0 for each score for the positive reviews. Thus, this model was likely not very helpful.

Creating the same model for the unfiltered review text, this model had an accuracy of 87.50%, which was slightly lower than the filtered text model. The 10-fold and 20-fold cross validation mean accuracies were both 88.10%. This was a slight increase from the accuracy of the base model, however still lower than the previous model. Again, when computing the precision, recall, and F-Measure scores, the values were 0 for each score for the positive reviews. Thus, this model was also not likely to be very helpful.

For a final model, I chose to run a Logistic Regression Model using SKLearn. Again, using the filtered text and the review star rating, this model resulted in an accuracy of 89.64%, which was the highest of all of the models run. Due to the simplicity of this model, it may not be the best to pick up the subtleties in the data, although it did have the highest accuracy score.

After creating six different models to analyze the text data from Amazon musical instrument reviews, I found that moderate text processing and a Naïve Bayes model were the most successful. The third model created had high accuracy scores, with and without cross validation, and had the highest precision, recall, and F-Measure scores. Though this model could likely be greatly improved, this would not be possible without the addition of more positive reviews in the data set, as it was very skewed towards negative reviews. For further analysis, I believe it would be beneficial if other models were created, such as random forest or neural network models, and if more data was gathered.

References:

Data Source: <https://www.kaggle.com/eswarchandt/amazon-music-reviews>

Code / Concept Assistance:

<https://www.geeksforgeeks.org/python-nlp-analysis-of-restaurant-reviews/>

<https://stackabuse.com/the-naive-bayes-algorithm-in-python-with-scikit-learn>

<https://medium.com/hugo-ferreiras-blog/confusion-matrix-and-other-metrics-in-machine-learning-894688cb1c0a>