Homework Assignment #3

BUILDING A REVIEW CLASSIFIER WITH ROTTEN TOMATOES

Bao Ngoc Dinh & Madeline Jordan
Web Analytics ISGB/BYGB 7978 | Fall 2019 | Prof. Apostolos Filippas
Gabelli School of Business Fordham University

Overview

For this project, we utilized our previously created parser from homework 2 in order to scrape data from 11 movies, which are listed at the end of this document, from the website Rotten Tomatoes. From there, we used this data to build a classifier which we then evaluated. Lastly, we performed exploratory data analysis on our data.

For each of the 11 movies we scraped a total of ten pages of reviews from each movie. From this we gathered information on a total of 2,089 reviews. Our data frame consisted of the following columns: Reviewers' Name, Rating (fresh or rotten), Review and Date. 'Review' was changed into 'string' type since it is a text review.

First, we attempted running the models without any type of pre-processing. We turned our documents into clean feature vectors then used the vectors to predict whether a review is fresh or rotten. This process included creating a vectorizer that tracked text as binary features, then let the vectorizer learn what tokens existed in the text data, then finally turned these tokens into a numeric matrix. We set 'Rating' as the Y – the data we are trying to classify and our vectorized comments as inputs for our model. The model returned an accuracy of 68.1%.

After creating this initial classifier, we tried several different models to see if we could increase the accuracy. Next, we attempted the popular method for normalizing term frequency, TF-IDF, or term frequency inverse document frequency. Through this, words count higher if they occur more frequently within that document and words are penalized for occurring frequently across documents. The model accuracy using this increased to 72.5%.

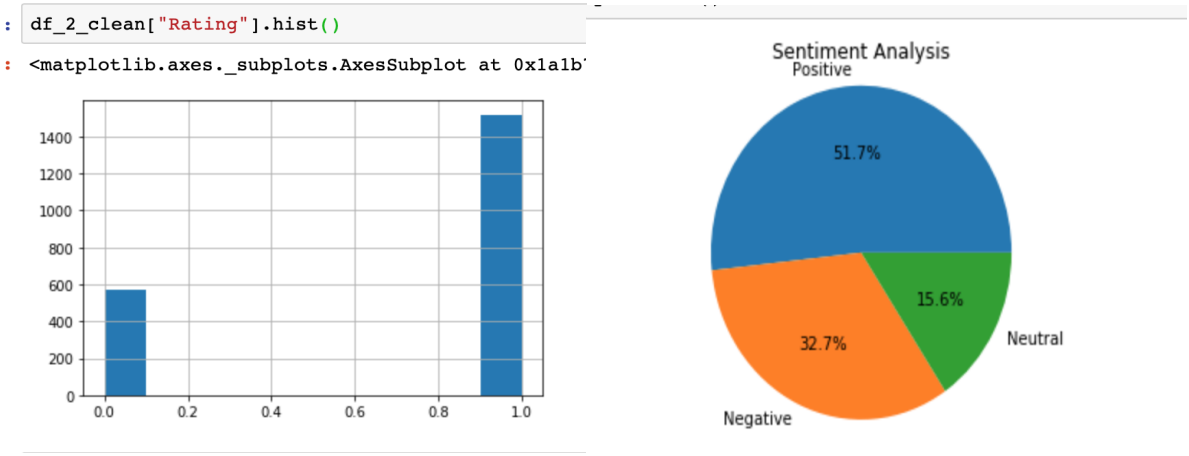
Next we tried the NTLK, or Natural Language Toolkit method, to identify stop words. Stop words are commonly used words like "a," "an," "as," "the," "am," "I," etc. We also changed all our words into lowercase by using lower() function. We applied this method over our

binary vectors method and the TF-IDF method. For the model using the binary vectors, the accuracy of our classifier actually decreased by 0.2%. However, the accuracy of the TF-IDF method increased by 0.01%.

Overall, we were able to successfully increase the performance of our classifier by utilizing several methods. Other methods that could be used includes using bi-grams instead of uni-grams, stemming and lemmatizing.

Exploratory Data Analysis

We created a histogram for the data collected. Overall, most of the reviews fell into 'Fresh' category (over 1,400 reviews) while the rest fell into 'Rotten' category (less than 600 reviews).



We used sentiment analysis to obtain more insights from the dataset. We compiled a dictionary of positive words and negative words from GitHub and loaded it into the python file. From there, we proceeded to count how many positive and negative words a review has. If the review has more positive words than negative words, the review is considered 'positive'. If the review has more negative words than positive words, the review is considered 'negative.' If the amount of positive words and negative words are equal, it is considered 'neutral.' Our sentiment analysis showed that 51.7% of the reviews is considered 'positive,' 15.6% is considered

‘neutral,’ and 32.7% is considered ‘negative.’ Sentiment analysis would be more useful if we can match the sentiment with the ‘fresh’ or ‘rotten’ ratings to see if sentiment analysis has a strong predictive power to predict rating outcomes.