Yelp Reviews Sentiment Analysis Using Natural Language Processing

Rabia Tariq – Data Scientist – Springboard School of Data

https://github.com/Rabia1995

October 18, 2021

1. INTRODUCTION

These days one of the most common ways customers use to share their thoughts on a product or a service is through online reviews. Most sites today allow users to give their opinion and feedback on products or services using text and a five-star rating system. The star system gives an overall general view of what the person thinks about the product or service, but if want to really know what that person is thinking then we should look at the test review. There is a huge amount of text data, and we can use that data in a productive way by classifying the data in the way we want. Through the text data, the companies can gain insights on the wider public opinion and act accordingly. With having so much data, it is very difficult for a human to read through all the reviews. So, there needs to be a better way to obtain valuable insights from the vast amounts of text data. For this reason, we can make a program that will read the reviews.

Sentiment analysis is contextual mining of text which identifies and extract subjective information in source material and helping a business to understand the social sentiment of their brand, product or service while monitoring online conversations. What might take a human a long time to read through a million lines of code, computer program can do it in a short amount of time. But even with the most complex models for sentiment analysis, they are not 100% accurate.

For this analysis, we will be using the text-based reviews from Yelp. Yelp is a site which allows its users to review different businesses and give a rating using the five-star rating system.

2. DATA COLLECTION

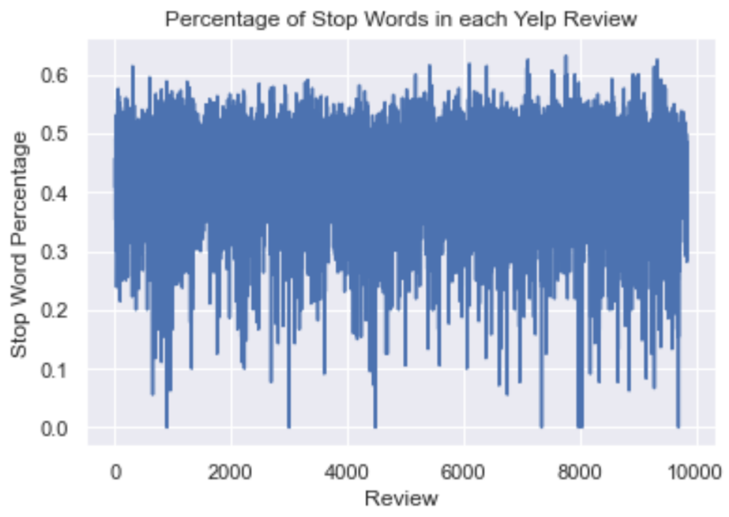
The data was collected directly from Yelp using web scarping. We collected almost 10,000 reviews from 197 New York City restaurants. We chose the restaurants from the NYC Open Data directory of restaurants. Then they were converted to valid URLs using Yelp's Fusion API.

The reviews were the stored as strings in a pandas data frame along with the name, rating, price, location and total number of reviews from each restaurant.

3. EXPLORATORY DATA ANALYSIS

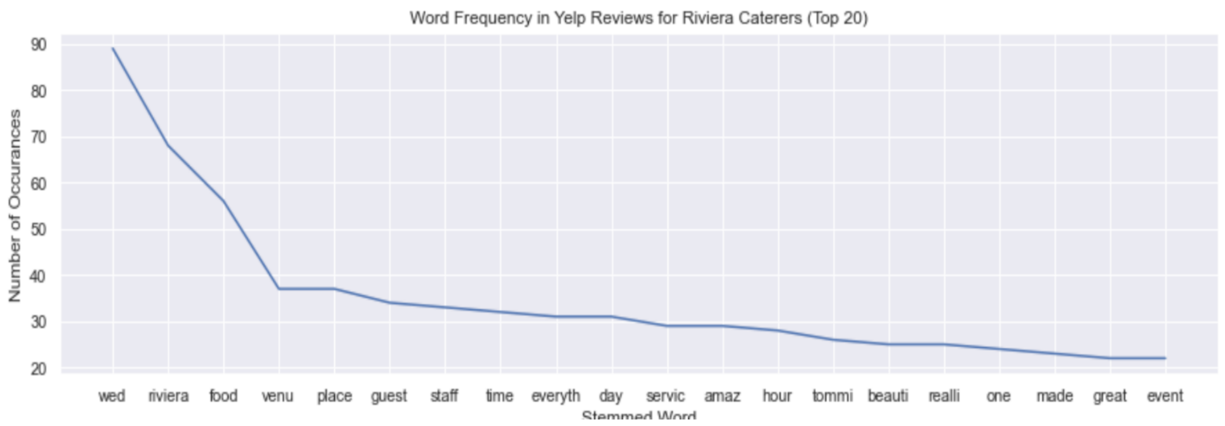
Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations. In our EDA, we also wanted to apply NLP techniques to see how this data will be used for the prediction model. As our reviews are text-based, we had to remove the stop words, tokenize each review, and apply stemming or lemmatization.

Stop words like 'its', 'an', 'the', 'for', 'and', 'that' don't add much for our sentiment analysis. Usually, these stop words are the most common in any text. So, we want to remove them to save memory and processing power and because we don't want these stop words to impact our sentiment analysis. So we remove them to ensure that the most frequent words in our reviews have some sort of sentiment.



In the above plot, we can see that on average most yelp reviews have about 40% stop words. So, removing these stop words will save a lot of time with our modeling.

Stemming is used to preprocess text data. The English language has many variations of a single word, so to reduce the ambiguity for a machine-learning algorithm to learn it's essential to filter such words and reduce them to the base form. In the areas of Natural Language Processing, we come across situation where two or more words have a common root. For example, the three words - agreed, agreeing and agreeable have the same root word agree. A search involving any of these words should treat them as the same word which is the root word. So, it becomes essential to link all the words into their root word. By doing this, we will make our dataset smaller and frequency analysis more accurate. So, by removing stop words and having them stemmed, we should be left with the words that are meaningful. Below is shown a plot that shows the top 20 words in one of the restaurants, Riviera Caterers.



Next, we convert the words into vectors. We will get a matrix where columns are each unique   word, and the rows are the total number of times that the word appears. Then we can compare   these word vectors and we can start seeing the numerical similarity between words. To find if the word or a phrase is positive or not, we can calculate its polarity value based on the TextBlob library's positive index.

Polarity is float which lies in the range of [-1,1] where 1 means positive statement and -1 means a negative statement. A polarity of 1 implies 100% positivity and polarity of -1 implies negativity. The TextBlob function takes each word vector in the review and gives a polarity score, then assigns a score for the review as a whole. This gives a way to classify each review and determine the sentiment.

We looked at another relation, which was the percentage of positive words in each review. We looked whether the reviews with high polarity have a noticeable increase in the percentage of positive words and vice versa for negative reviews. By doing this, we will make our model more accurate. We do this by using Positive Opinion Lexicon, which is a collected list of over 200 known English positive words. By iterating through each review, we compare each word to the word in the Lexicon list and then divide the number of positive words by total word count to get the percentage of the positive words.

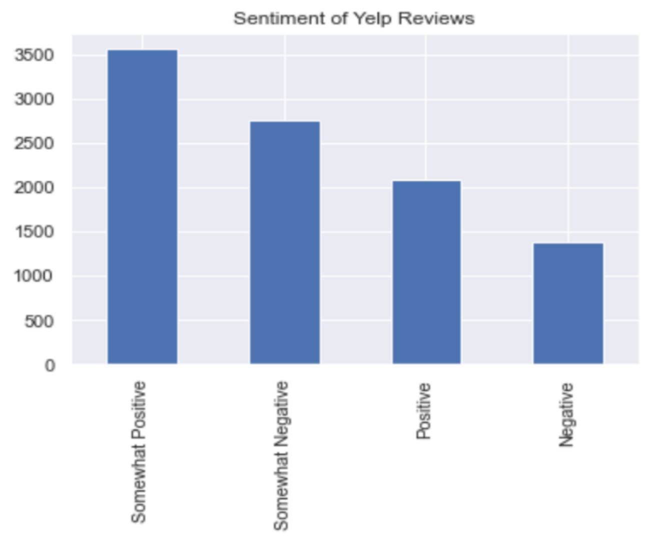
We confirmed that the reviews with higher polarity have a higher percentage of positive words. So, after all this work our dataset was ready for preprocessing.

4. PREPROCESSING

Now, our next step in performing sentiment analysis is preprocessing. In order to create a model that will accurately assign each review to a sentiment class, the model needs to be able to process two different kinds of data:

- Number based, like positive word percentage or classification indices.
- Text-based, like word matrices or Term Frequency Inverse Document Frequency (TF-IDF) vectors.

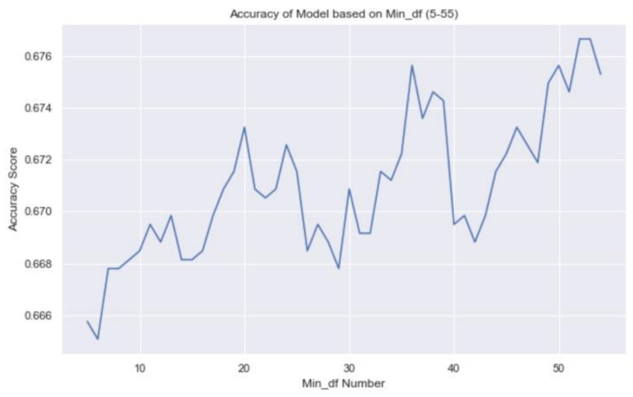
Firstly, as our data is unlabeled, we labeled our reviews as positive or negative. Most of the reviews in our dataset were positive, so our dataset was kind of unbalanced. But according to Yelp, most Yelp community members usually write more positive than negative reviews. So, it is expected. But to make the data more balanced, we created 4 classes that the data can be classified into. They were ‘Positive’, ‘Somewhat Positive’, ‘Negative’, ‘Somewhat Negative’. This can be seen in the plot below. To prepare the dataset to be converted into train/test split, we need to select the right features to not under or overfit the model.

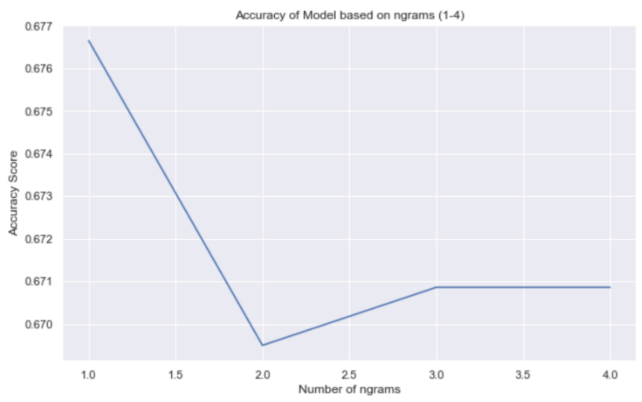


Through some analysis, it seemed that percentage of positive words would be the best for training the model. The percentage of stop words was around 40% for each category and the stemmed word count was very similar three of the four groups. However, for the average percentage of positive words are distinct for each class which could be used to improve the model's predictions.

Next, we select the text-based features. In order for our model to be able to interpret the text from each review, we need to convert the text into a matrix using CountVectorizer or TFIDFVectorizer. For our model to have the best performance, we will be testing out both. It is also important to determine the ideal number of ngrams (tokens or token pairs) and min/max document frequency for the TFIDFVectorizer.

To test each feature, a function was written to run tests on a simple logistic regression to see which set of features would return the highest accuracy score. The testing pipeline tested TFIDF vs CountVectorizer first, then minimum document frequency, and finally the number of ngrams.





In the end, the best result was TFIDFVectorizer with a minimum document frequency of 52 and 1 ngrams for each token. This means that the model will only take words that have appeared 52 or more times throughout our ~10,000 reviews. This will help with overfitting and saves a lot of time and processing power. Finally, we can export a processed dataset that has the percentage of positive words and a TFIDF vector for each review. Now, we can create a model.

5. MODEL SELECTION

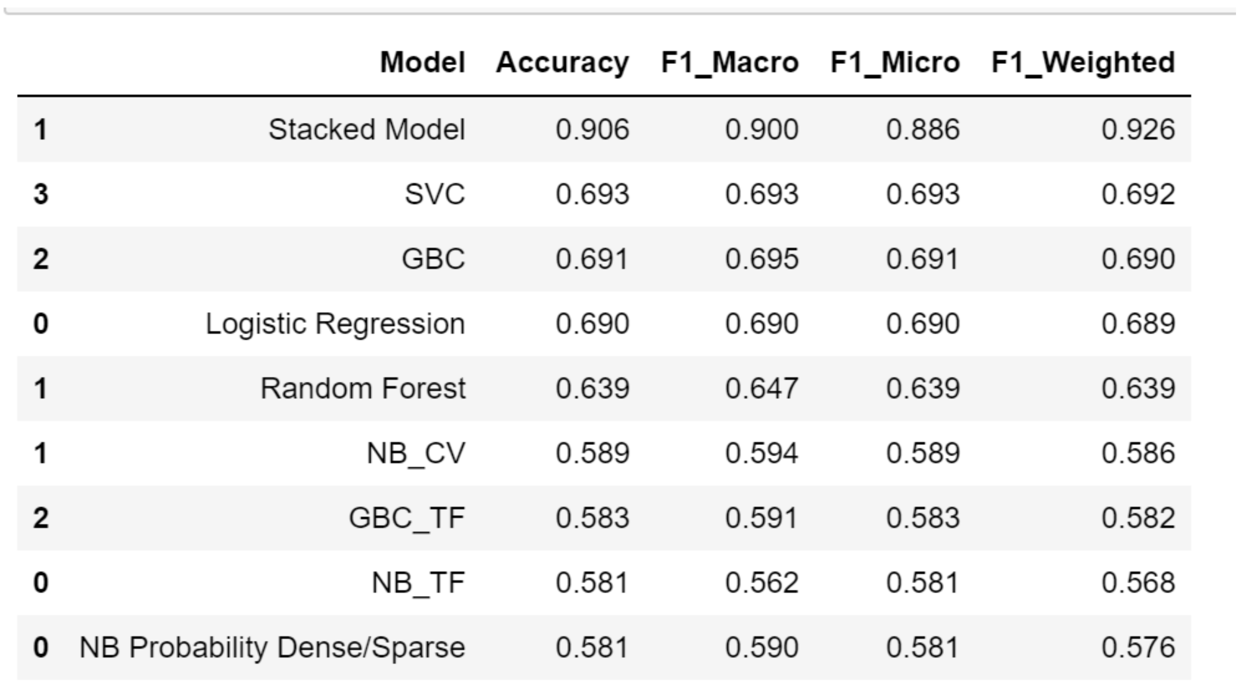
Up till now, our baseline logistic regression model with ideal feature parameters has an accuracy of 68%. There is room for improvement using other modeling techniques. For our sentiment analysis, the model testing process will look like this:

- Select model type.
- Define a pipeline and use GridSearch to find the best parameters.
- Save the model, accuracy, and F1 scores.
- Compare the results.

To get a good variety of results, we tested different types of models including, Logistic Regression, Random Forest, Gradient Boosted Classifier, Support Vector Classifier, text-based Naïve Bayes, text-based Gradient Boosted Classifier, Naïve Bayes with dense and sparse features and a stacked model. We were able to find that the stacked model had the best performance and was the most accurate with 90% accuracy. We achieved this by combining the probability of the Naïve Bayes model with the feature performance of the Gradient Boosted Classifier.

We created this stacked model by converting the full review text into a sparse matrix and then into a dense matrix. Matrices that contain only zero values are called sparse, and those that contain most non-zero values are called dense. Using the dense matrix as an input and the classification as an output, the Naïve Bayes model outputs the probability value based on the content of each review. This number can the be added to the training dataset that was created in pre-processing to make an improved training dataset.

Then Gradient Boosted Classifier was used on the new combined dataset. This stacked model combines the best text-based model with the best numerical model to improve our baseline accuracy by over 20%.



	Model	Accuracy	F1_Macro	F1_Micro	F1_Weighted
1	Stacked Model	0.906	0.900	0.886	0.926
3	SVC	0.693	0.693	0.693	0.692
2	GBC	0.691	0.695	0.691	0.690
0	Logistic Regression	0.690	0.690	0.690	0.689
1	Random Forest	0.639	0.647	0.639	0.639
1	NB_CV	0.589	0.594	0.589	0.586
2	GBC_TF	0.583	0.591	0.583	0.582
0	NB_TF	0.581	0.562	0.581	0.568
0	NB Probability Dense/Sparse	0.581	0.590	0.581	0.576

6. CONCLUSION

Now, we have our trained stacked model which can be used in the future. We can see that we started with a baseline accuracy of 68% with Logistic Regression. We then used different models, including Random Forest, Gradient Boosting Classifier, Support Vector Classifier, Naive Bayes and Stacked Model. Some models performed better than our baseline model, but some performed worse.

With our stacked model we were able to improve the accuracy from 68% to 90%, which is a 22% increase in the accuracy. We can now use this model to classify new reviews. The model accuracy will only improve over time as more data is tested. This has major business implications and could be very useful for any service that receives mostly test-based reviews. We can change the classification groups and what is being classified and answer different questions using similar NLP methods. The potential for natural language processing is incredibly high. This sentiment analysis shows just one of the many applications of utilizing text-based data.

.