Sentiment Analysis on Yelp Challenge Data

Giselle Kurniawan[‡] (UID: 905571764)
Kyle Lee[†] (UID: 205848474)
Joyce Mok[†] (UID: 505343038)
Isabelle Supandji[†] (UID: 505400335)

[‡] Department of Statistics

1 Introduction

Natural Language Processing also known as NLP is a subfield of artificial intelligence concerned with how computers process natural language data. Our project aims to use NLP on a large Yelp reviews data set to correctly predict a user's rating of a business on Yelp on a 5-star rating scale.

The 5-star rating scale is a popular and well used by many platforms such as Google Reviews, Yelp and even educational rating websites like BruinWalk and Rate My Professor. Though sound in theory, objectivity is not always present. An example of this is that a rating of 4 stars may have different meanings to different people. To some it might just mean "above average" while others may view it as "very good". Thus, many rely on written reviews to understand the rationale behind ratings.

Using a Yelp data set with over 53,845 observations, we seek to train models that will predict a user's star rating of a business based on the positive and negative language used in their reviews. As computers cannot process language in the same way humans do, our challenge in this project is to program code that can accurately predict the sentiment of the review. Through processing the data, selecting predictor variables and comparing models, we aim to train our program to process language to correctly predict the user's rating.

To implement text classification, we have specified several well-defined steps that are depicted in Figure 1 and explained in more detail throughout the paper.

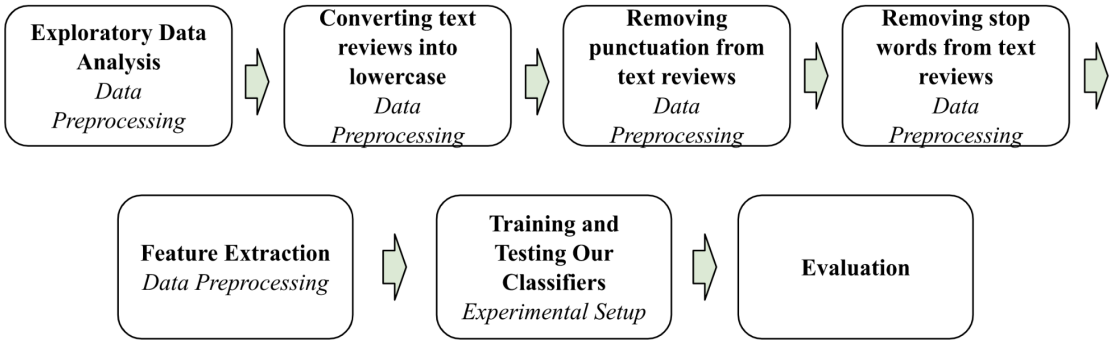

Figure 1: Text Classificationn Steps

2 Pre-Processing Step

To ensure that our machine learning model is built on quality data, we need to do some data clean up and pre-processing. To do this, we conducted exploratory data analysis to see if we can find trends or patterns that might be helpful when designing our model. To deal with potential outliers, the first metric we looked at was the word count for each review.

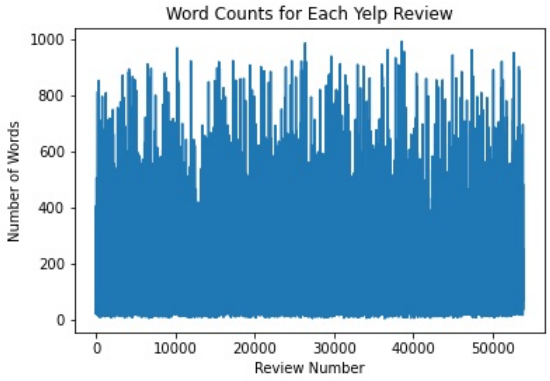

Figure 2: Bar Graph of Total Word Count for Each Review

From the plot above, it seemed like most reviews are under 1000 words, so we did not have to remove any outliers.

Furthermore, we wanted to see the distribution of reviews for each of the 5 ratings and we found that the majority of reviews leaned positive in the 4 to 5 star range, while negative reviews in the 1 to 2 star range had the least representation. From this discovery we are able to accommodate for the disproportionate densities in observations in the dataset when analyzing the accuracy of our models.

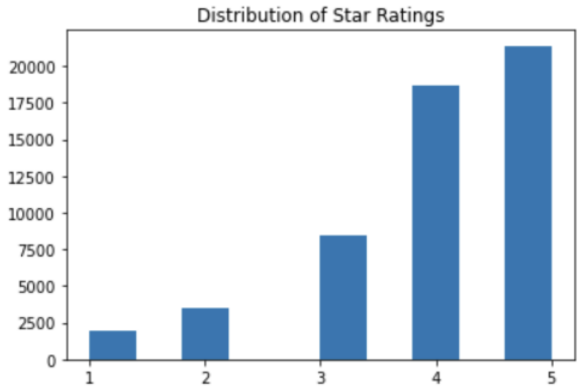

Figure 3: Bar Graph of Distribution of Star Ratings

In order to perform sentiment analysis on high quality data, we decided on several steps to improve overall efficiency and to make the overall meaning of each review easier to understand with less filler:

1. Standardizing all letters into lowercase
2. Removing any punctuations
3. Removing stop words such as 'the', 'a', 'for', or 'so' that bring less meaning for sentiment analysis.

After completing these steps, we created a plot to see whether removing the stop words made a significant impact on the overall meaning of each review. We assumed that the stop words to total word count ratio per review would be a good indicator of overall meaning.

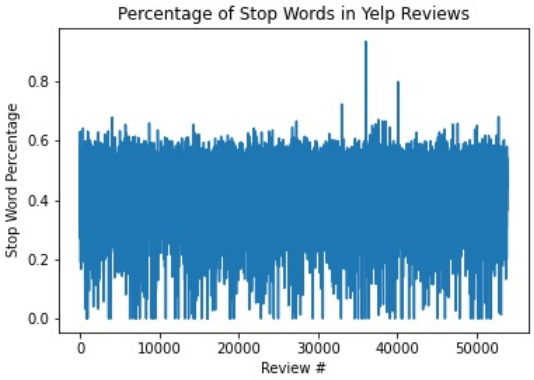

Figure 4: Bar Graph of Percentage of Stop Words in Each Review

According to our plot, stop words accounted for 40% of a review on average. We believe that this value is pretty significant and therefore decided that removing the stop words made

a significant impact on our dataset.

Now that we have cleaned our reviews column, we then dropped unnecessary columns from our dataset to make model building easier. This resulted in a clean dataset with columns: "User_id", "Bus_id", "Star" and "clean_review".

Feature Extraction
Since machine learning models are unable to work with raw text directly, we need to convert our review column into numbers that can be fed into our model.

We came up with two ways to convert our raw text review into numbers:

1. Generating a score for each review using the lexicon files of positive and negative words provided in Bruinlearn

2. Using Bag-of-words and TF-IDF to transform raw text into numerical feature vectors

For the first method, we counted the number of words in a review that were also in the lexicon files provided. This resulted in two additional columns in our dataset, positive_to_total and negative_to_total, to indicate the positive to total word count ratio and negative to total word count ratio in each review, respectively.

For the second method, we used the popular bag-of-words (BoW) method to transform raw text into numerical feature vectors with CountVectorizer. With CountVectorizer, we tokenized raw text into a dictionary of features and a term-document matrix based on the frequency (count) of each word that occurs in the entire text.

3 Experiment

As the goal of our study was to predict the star rating of a Yelp review given the words used in the review, the first step we took was identifying what variables should serve as our predictor variables. Using PCA we found that the two predictors that explained the most variability in yelp ratings was the ratio of positive words to total words and the ratio of negative words to total words. Using these two predictor variables we split the data into training and testing data sets at an 80:20 ratio. Then we used the training data to fit the various models we created and analyzed each model's accuracy by calculating accuracy scores and creating confusion matrices to visualize the accuracy. Thus, we created two types of models: binary classification and multinomial classification.

First, we used binary text classification to separate user reviews into two general classes: positive and negative. For our second approach, we concentrate on a deeper analysis where user reviews are classified into five different classes, corresponding to the existing five-star

rating scale.

For these two approaches, different classifiers will be tested and evaluated through an accuracy score.

Binary Text Classification In this approach, the goal is to predict the general sentiment in Yelp user reviews. To conduct binary classification, we decided that logistic regression would be a good method as it accurately predicts the probability of a categorical dependent variable. Hence, we built a logistic regression model that predicts whether a rating is positive (1) or negative (0).

To attain accurate results, we only took reviews with polar opposite ratings:

- Positive: Star = 5 (class 1)
- Negative: Star = 0 (class 0)

Since we filtered our dataset to only 1-star or 5-star ratings, this resulted in 23,285 reviews. Generally, reviews belonging to the POSITIVE review category will positively influence a consumer's decision to purchase a product, while reviews belonging to the NEGATIVE review category will tend to discourage consumers from buying a specific product. Using Python's Scikit-learn library to split our dataset into 80% training data and 20% testing data, we trained our classifier and resulted in a 94.4% accuracy score. Though our classifier performed extremely well in predicting whether a review is positive or negative, we wanted to conduct a deeper sentiment analysis that classifies reviews into a 5-star rating scale.

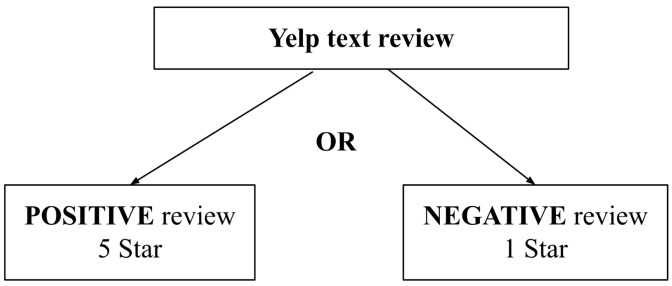

Figure 5: Binary Classification Representation

Multinomial Classification
To refine our analysis, we decided to extend binary classifiers to multinomial (multi-class) classification problems. With binary classification models, we were only able to predict whether a review was positive or negative. In multinomial classification, we could predict the star rating of a review on the one to five rating scale. Note that classes are mutually

exclusive such that each instance is assigned to only one class. For instance, a text review cannot have both a 1-star and 2-star rating.

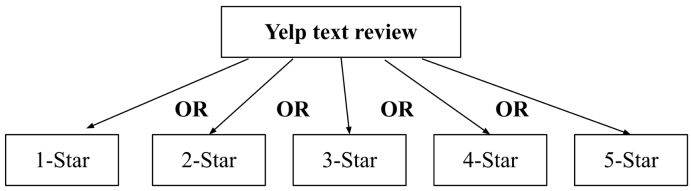

Figure 6: Multinomial Classification Representation

Training a Multinomial Classifier

In the training step, we feed class-review mappings from our training data to the classifier. This enables the classifier to analyze patterns in these instances and later, accurately classify new instances. With Python's Scikit-learn library, we used the train_test_split method to split our dataset into 80% for training data, and 20% for testing data. In order to find the best multinomial classifier for our dataset, we decided to train five different classifiers: K-nearest neighbors (KNN), Decision Tree, Naive Bayes, Random Forest, and Logistic Regression.

To find the best classifier for our dataset we first trained the model and analyzed and compared their accuracy scores against each other. Accuracy scores refer to the ratio of correctly predicted ratings by the classifier against the true rating values.

K-nearest neighbors (KNN)

The Decision Tree is a non parametric supervised machine learning model that uses a set of rules to make decisions. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

From our first trial at a KNN model, we received the following confusion matrix with an overall accuracy rate of around 35%:

Due to our low KNN accuracy, We wanted to further visualize our data in order correctly and efficiently choose the correct model that would best fit our data. To do so, we created a 2D scatter plot of the data labeled by their 5 star rating.

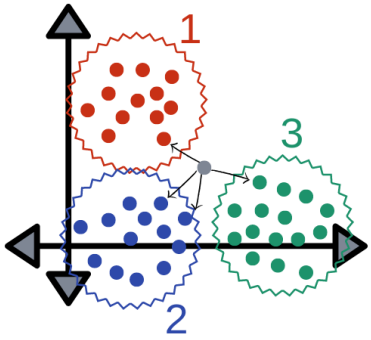

Figure 7: Visualization of how KNN works

(a) source: https://machinelearninghd.com/k-nn-k-nearest-neighbors-starter-guide/

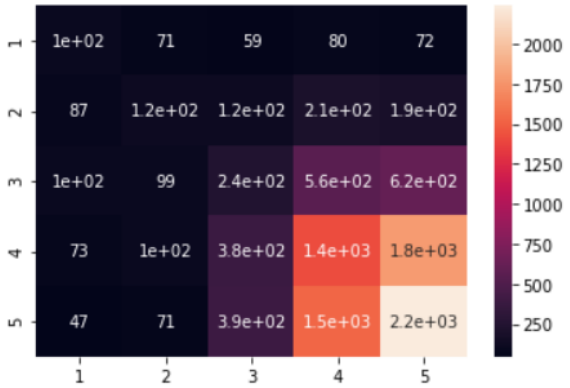

Figure 8: Heat Map of initial KNN Confusion Matrix

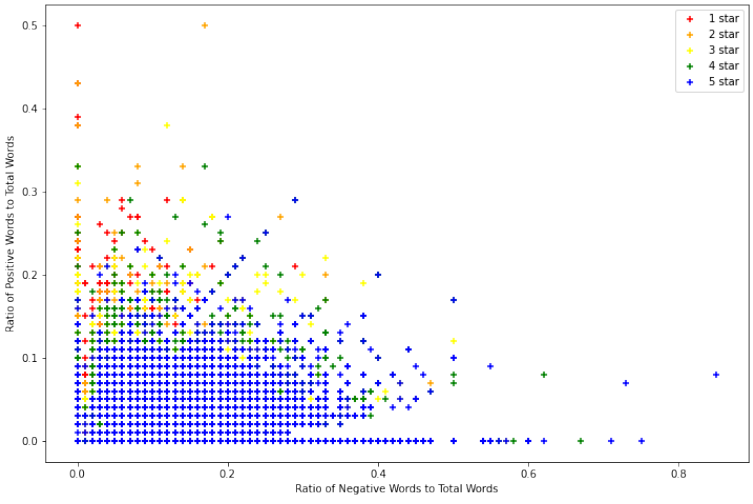

Figure 9: 2D Scatter Plot of Positive Word Ratio to Negative Word Ratio

From the above plots, we can see that the plot of our data is not fitting for the KNN model. The ideal data set for the KNN model would have data points of the same classification clustered in different areas of the plot (as seen in figure 5 below). This is due to the fact that KNN is calculated by Euclidian distance. However, our data has points without distinct areas for different classes. This is seen by the fact that most points regardless of their class are clustered near the origin. Thus, no clear distinction can be made between the classes based on their distance from one another.

Naive Bayes Multinomial Classifier

After doing some research on NLP methods, we decided to implement a Multinomial Naive Bayes classifier as it is a popular method for text analysis. Multinomial Naive Bayes is an algorithm based on Bayes Theorem, and predicts the class of a text review. In specifics, it calculates the probability of each class for a given sample and then gives the class with the highest probability as output.

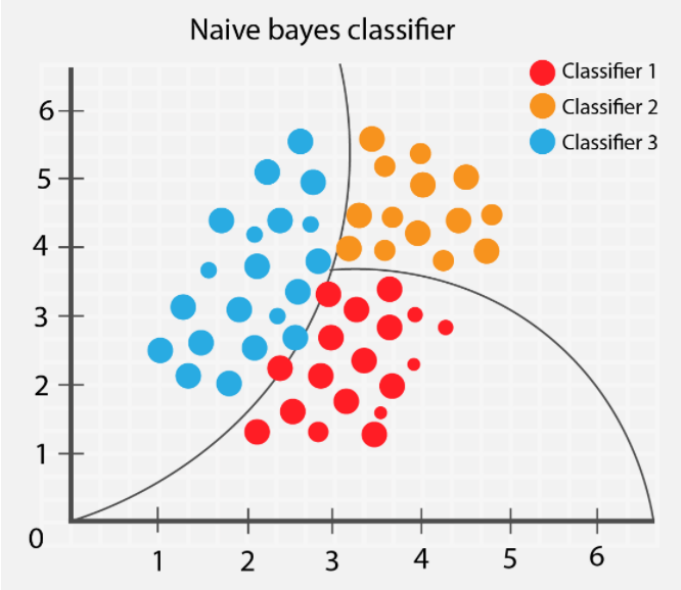

Figure 10: Visualization of how Naive Bayes Works

(a) source: https://kdagiit.medium.com/naive-bayes-algorithm-4b8b990c7319

Decision Tree Classifier

The Decision Tree is a non parametric supervised machine learning model that uses a set of rules to make decisions. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

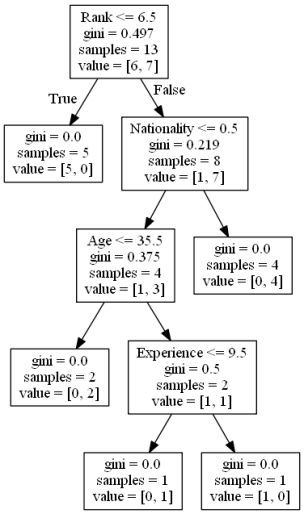

Figure 11: Visualization of how Decision Tree Works

(a) source: https://www.w3schools.com/python/python_ml_decision_tree.asp

Random Forest Classifier

The Random Forest model that uses repetitions and folds to create a 'forest' of decision trees, where each decision tree is trained using a random subset of features. Random Forest uses a voting system and weighted combination to decide a final prediction based on the individual prediction of the decision trees.

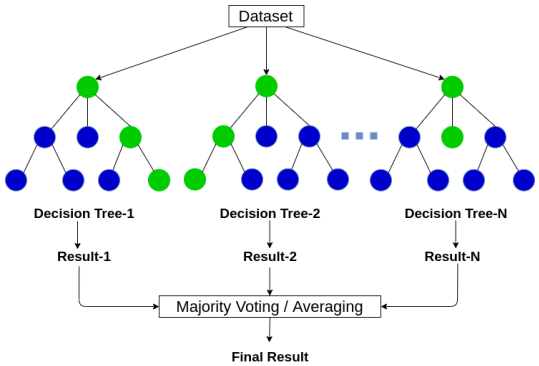

Figure 12: Visualization of how Random Forest works

(a) source: https://medium.com/@curryrowan/the-complete-guide-to-random-forests-part-2-934eabf35534

Logistic Regression

The Logistic Regression model is a parametric model that estimates the probability by taking the log-odds for the event to be a linear combination of at least one independent categorical variable.

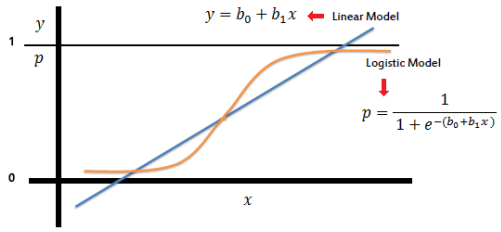

Figure 13: Visualization of how Logistic Regression works

(a) source: https://kdagiit.medium.com/naive-bayes-algorithm-4b8b990c7319

4 Analysis

After building various models and preprocessing the data with two different natural language processing approaches, we analyzed the accuracy of each model to find the model that was able to best predict the star ratings of a user's review through the sentiment of the words in the reviews. Looking at both accuracy scores and confusion matrices, we were able to find the best model for predicting our target variable: user star rating. Our accuracy score is the ratio of correctly predicted user star ratings compared to the true predicted user star ratings.

Binary Classifiers:

In our binary classification approach, we attempted to use a machine learning model to predict whether a review was given a positive (4 to 5 star review) or negative (1 to 2 star) review. Ultimately, we found that the Logistic Regression model was the most accurate model given that it had an accuracy score of 94.4% when using the positive to total and negative to total ratios as predictor variables to predict our target variable of user ratings.

Multinomial Classifiers (Lexicon File Approach):

Using the given lexicon files of positive and negative words to calculate a positive to total ratio and a negative to total ratio, we found that models built using these predictor variables to predict user star ratings were less accurate than our second approach. We built 5 multinomial classifications models (KNN, Decision tree, Naive Bayes, Random Forest, and Logistic Regression) using this natural language processing method and found that our models were relatively inaccurate. The model with the highest accuracy under this preprocessing

approach was the Decision tree model with an accuracy score of 43%. Essentially all models using this pre-processing approach scored equal to or below 43% accuracy. The exact accuracy scores of each approach are displayed in the table below.

Multinomial Classifiers (Bag of Words and TF-IDF Approach):

Using a combination of the bag of words and TF-IDF to do natural language processing, we found that models built using this approach were significantly better in accuracy than the previous pre-processing approach. Using the same five multinomial models in the first approach (KNN, Decision tree, Naive Bayes, Random Forest, and Logistic Regression), we built these models using this preprocessing method and found that every model improved in accuracy compared to the first pre-processing approach. However in this approach, the model that provided the highest accuracy score was the Logistic Regression model at 57% accuracy rather than the Decision Tree model like in the first approach. Thus, these improvements in model performance indicate that the natural language processing approach greatly impacts the ability of a model prediction.

Feature Extraction:

1. Lexicon Files

APPROACH	CLASSIFIER	ACCURACY SCORE
Binary	Logistic Regression	94.4
Multinomial	KNN	32
	Decision Tree	43
	Naive Bayes	37.7
	Random Forest	40.8
	Logistic Regression	42.9

Figure 14: Accuracy Score of Models Using Lexicon Derived Variables

2. Bag of Words and TF-IDF

APPROACH	CLASSIFIER	ACCURACY SCORE
Multinomial	KNN	43.2
	Decision Tree	43
	Naive Bayes	51.9
	Random Forest	53.3
	Logistic Regression	57

Figure 15: Accuracy Score of Models Using Variables Derived with Bag of Words and TF-IDF

5 Conclusion and Summary

Ultimately, we sought out to build a suitable model that would allow us to utilize language processing techniques to parse the text into two categories of words, words with positive and negative sentiments. Via different model comparisons we found that KNN was a suitable model for our data. In future iterations of this project we look to do model comparisons to obtain the best model fit to predict yelp ratings. These were some of our main findings.

With the multinomial classification approach, we were able to build 5 different classifiers that predict the exact star rating of a Yelp review. Amongst these multinomial models, the best performing model was the Bag of Words and TF-IDF Logistic Regression Model with an accuracy of 57%.

Across the board, the Bag of Words and TF-IDF approach created more robust models with higher accuracy than that of their counterpart that utilized Lexicon files. This was present in all 5 models we created, ex. the KNN model using Bag of Words and TF-IDF had a higher accuracy than the KNN model using Lexicon files. This can be explained by the fact that Bag of Words and TF-IDF are more flexible and adaptable to the data. The Lexicon approach involves matching words to a definitive list of negative and positive words. Thus, the number of positive and negative words are used to create the predictor variables. However, the Bag of Words and TF-IDF approach collected information from the data set by counting the frequency of each word in each review and using Term Frequency and Inverse Document Frequency as measures of how important and positive/negative leaning a word is. Thus, the flexibility and adaptability of this feature selection method was more suitable to our dataset, generating a higher accuracy score.

Among the 11 models we created, the model that had the highest accuracy was the Binary Logistic Regression with 94.4% accuracy. Though this is close to perfect, this model is binary and only predicts between positive and negative. Thus, it is less complex and does not provide as accurate of a rating. However, among as mentioned earlier, the logistic regression using Bag of Words and TF-IDF provided the best accuracy at 57%.

With this, we believe we have done in-depth exploration, pre-processing steps, and modelling on our Yelp data set. However, if we were able to delve further into machine learning and natural language processing methods, we would explore other multinomial classifiers like Support Vector Machine (SVM) and Neural Networks that are known to produce higher accuracies. Other possibilities we would explore include stacking different models that combine text and numerical based data to see whether this would increase the accuracy significantly. An example of this would be combining the Naive Bayes Classifier with Gradient Boosted Classifier.

We hope that our research and study can provide Yelp customers with a more accurate

rating for each restaurant, which could assist in decision making of which store or restaurant one would visit next.