Classification of Yelp's Star Ratings based on Text Reviews

Ramiro Romero
Department of Statistics, UCLA

Submitted to Stats 101C Date: 12/02/22

1.0 INTRODUCTION

I. Background

Yelp is a popular online directory for discovering local businesses ranging from bars and restaurants to salons and café's. Because of its popularity, the majority of American businesses are encouraged to list themselves on yelp where users can rate a business from 1-5 stars and justify their rating with a text review. With the growth of businesses available for review on Yelp, the online directory expanded into an online community where users can interact with each other. Users can rate each other's text reviews from 1-5 stars on 3 of several features; funny, cool, and helpful. With the utilization of a large open dataset, we hope to implement several classification techniques to predict whether a text review is positive or negative. I will consider ratings of 4 and 5 stars as positive, and ratings of 3 stars or less will be considered negative. This process of implementing a machine learning algorithm to classify text is referred to as sentiment analysis or sometimes referred to as opinion mining. This process can be extended to many other applications where the assessment is in the format of text and the goal is to assign a level of some kind of feature. Some examples include the detection of plagiarism in student papers or finding inappropriate video game lobbies based on game chats or audio scripts.

II. Goal and Outline

The goal of this project is to apply existing supervised learning techniques to predict the sentiment of a review based on text alone. We utilize the yelp dataset provided through the yelp dataset challenge. It is an open dataset which is accessible by anyone for the purpose of academic research. I experiment with two machine learning algorithms including random forest and logistic regression. I compare the performance of two ML algorithms on the basis of three evaluation metrics; precision, recall, and f1 scores. All of the code is executed in Rstudio and focuses primarily on text parsing using the tm library which provides a framework for text mining applications in R.

2.0 DATA

I. Data Reduction

The original data frame is exceptionally large, consisting of 53,845 observations of 18 variables. R is a single threaded language which means that it is significantly slower when performing computations on such a large dataset. Because of the computational setback, I decided to focus solely on text reviews from the city of Santa Barbara, which accounts for nearly 80 percent of the original data and reduces our number of observations by over 10,000.

This project focuses solely on sentiment analysis using text review. Therefore, columns that aren't of interest are removed from the dataset so that the only variables left are a logical factor indicating whether a Text is positive or negative and the text itself. Ratings greater than or equal to 4 stars are considered positive and otherwise negative.

Finally, I am left with a data frame which consists of 42,532 observations and 2 variables; text review, and a factor indicating whether or not the text is positive.

II. Text Mining

Utilizing the text mining package in R, which can be accessed at [1], I processed the text into a format which is suitable for the ML algorithms. I first create a corpus which is metadata describing the text reviews. Using several text parsing functions I transform all characters into lowercase, remove numbers, remove punctuation, remove stop words, and I strip white space. I also put the text through a technique called stemming which lowers inflection in words to their root forms. Words like "loved", "lovely", and "loving" for example are reduced to their root form "love". Finally, I format the text into a matrix such that each row represents an individual text review, each column represents a word, and each entry represents the frequency of a given word in a specific review.

III. Bag of Words

I create a sparse matrix and set the sparsity to 0.99 which returns to me the top 99% of words based on the frequency of their occurrence. Then, I add a column called "Positive" that is a logical vector indicating whether or not a review is positive. After this process, I am left with a review sparse matrix with 42,532 reviews represented as rows, 1,006 words represented as columns, and a dependent variable "Positive" represented as the 1,007th column.

IV. Training - Testing split

The review sparse matrix is randomly split so that 80% of observations are used to train ML algorithms and 20% of observations are used to test their performance.

3.0 SENTIMENT ANALYSIS

I. Random Forest

Using 10 trees and 31 variables tried at each split, I trained a random forest algorithm to predict whether or not a review is positive based on text alone. Then, I evaluated the algorithm's performance using the testing data. The following confusion matrix and classification metrics show the performance of the random forest on the testing data.

Table 1.1

	Reference	
Prediction	False	True
False	874	459
True	1233	5939

Table 1.2

Precision	Recall	F1
0.8280814	0.9282588	0.8753132

The information in table 1.1 and table 1.2 shows that the algorithm performs well when classifying positive reviews, but it poorly classifies negative reviews. The false positive rate of the random forest is high, which is an indication that the training dataset is biased towards positive reviews.

The classification metrics show that overall the model is performing well, with an overall F1 score of 87.5%.

II. Logistic Regression

Logistic regression estimates the probability that a review is positive, so training a LR model will return values between 0 and 1. However, because the dependent variable is a logical vector the output from LR must be coerced into a binary class consisting of true and false values. To do this, a threshold of 0.5 should be imposed on the output from LR so that each value less than 0.5 is coerced into false and each value greater than or equal to 0.5 is coerced into true. After coercing the output from LR, I tested the performance of the model on the testing data.

Table 2.1

	Reference	
Prediction	False	True
False	1243	369
True	864	6029

Table 2.2

Precision	Recall	F1
0.8746554	0.9423257	0.9072305

The logistic regression model is performing well on the testing data, with an exceptionally high score for recall. The confusion matrix also shows a slight indication of bias towards positive reviews in the training data.

4.0 RESULTS & FURTHER ANALYSIS

The logistic regression model performs better than the random forest. This is because logistic regression exhibits higher performance when the number of nuisance variables are less than the number of explanatory variables. Because the model was trained using the frequency of 1,006 words, it is safe to assume that the number of explanatory variables is greater than the number of nuisance variables. Under the same conditions however, random forest exhibits a high true and false positive rate, explaining the high false positive rate of the random forest.

Further analysis can be done to find optimal parameter values for the random forest to increase its performance. However, given the conditions of the data, the model will likely still exhibit a natural bias towards positive samples.

Because I wanted this paper to focus specifically on text parsing and ML techniques, I didn't address the imbalanced training data set. One method to decrease positive bias is by removing positive samples, however this risks decreasing the overall accuracy and F1 score of the model. Another preferable method is to attribute weight inversely proportional to the frequency of classes.

ACKNOWLEDGMENTS

The author of this report would like to thank his classmates Fion Ho and Garren Lee who were helpful in the discussion of potential techniques and methods for model enhancement. The author would also like to thank Professor Shirong Xu and TA Joseph Resch for their help in the data collection as well as their helpful recommendations regarding the steps of the data analysis.

REFERENCES

[1] G. Ganu, N. Elhadad, and A. Marian, "Beyond the Stars: Improving Rating Predictions using Review Text Content.," WebDB, no. WebDB, pp. 1–6, 2009.

[2] N. Godbole, M. Srinivasaiah, and S. Skiena, "Large-Scale Sentiment Analysis for News and
Blogs.,"ICWSM, 2007.

[3] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up?: sentiment classification using machine learn- ing techniques," Proceedings of the ACL-02 con- ference on Empirical methods in NLP, 2002.

[4] S. Zanan, "Classification of reviews using methods of NLP and Decision tree algorithm.," RPubs, 2019.