Sentiment Analysis of Yelp ‘s Ratings Based on Text Reviews: A comparison of different classifiers and their accuracies

Neel Dave, Rangita Rajakumar, Syed Abrar Ahmed
University of Central Florida
{neelkdave, rangita1235, abrar.syed860}@knights.ucf.edu

Abstract

Yelp has been one of the most popular sites for users to rate and review local businesses. Businesses organize their own listings while users rate the business from $1-5$ stars and write text reviews. The rise of powerful algorithms that can perform sentiment analysis has instant and powerful applications. The application of sentiment analysis is focused on Yelp restaurants reviews in this paper. The goal of our project is to apply existing supervised learning algorithms to predict a review ‘s rating on a given numerical scale based on text alone. We look at the Yelp dataset made available by the Yelp Dataset Challenge. We first tokenize and vectorize the dataset using Scikit-learn. We also visualize the statistics of the stars using Seaborn library. Then we experiment with different machine learning algorithms such as Multinomial Naive Bayes, K-Nearest Neighbor and Support Vector Machine and compare our predictions with the actual ratings. We develop our evaluation metric based on precision and recall to quantitatively compare the effectiveness of these different algorithms. Finally, we compare the accuracies of these algorithms.

1 Introduction

Sentiment analysis is the process of determining whether a text is positive or negative. It is also known as opinion mining, deriving the opinion or attitude of a speaker. Sentiment analysis mainly follow keywords which represents positive and negative polarity. There are many ways that people analyze bodies of text for sentiment or opinions, one of those methods is using Natural Language Processing, and the attempt to truly "understand" the text. Usually, this structure requires the machine to understand grammar principles. To do this, Natural Language Processing (NLP) techniques are used to tag parts of speech, named entities, and more, to understand the "language" of the text, and not just look for target words. These techniques have major impact on many applications. One of them is Yelp. Yelp data is reliable, up-to-date and has a wide coverage of all kinds of businesses. Millions of people use yelp and empirical data demonstrated that Yelp restaurant reviews affected consumers' food choice decision-making. With the rapid growth of visitors and users, we see great potential of Yelp restaurant reviews dataset as a valuable insights repository. As increasing number of customers rely on Yelp for food hunting. Therefore, the review on Yelp has become an important index for food industry. Using this enormous amount of data that Yelp has collected over the years, it would be meaningful if we could learn to predict ratings based on review’ text alone, because free-text reviews are difficult for computer systems to understand, analyze and aggregate. Three machine learning algorithms have been used to train our model to evaluate the sentiment from the review. Before we train out model we first tokenize and vectorize the dataset using Scikit-learn. We also visualize the statistics of the stars using Seaborn library. We then experiment with three different machine learning algorithms, Multinomial, K-nearest neighbor, and SVM to train the model. Finally, we compare the accuracies of these three models with actual ratings. We develop our evaluation metric based on precision and recall to quantitatively compare the effectiveness of these three different algorithms.

2 Background

Related work has been in the paper Sentiment Analysis of Yelp's Ratings Based on Text Reviews.Yun Xu, Xinhui Wu, Qinxia Wang. Using the enormous amount of data that Yelp has collected over the years, it would be meaningful if we could learn to predict ratings based on review ‘s text alone, because free-text reviews are difficult for computer systems to understand, analyze and aggregate. We replicate it, but we are using different algorithms for predicting the reviews. Also, we extended it by comparing the accuracies of the three algorithms and conclude which has best accuracy. And in doing so we get more better accuracy than the previous work done in this paper.

3 Data Source and Preparation

The data was downloaded from the Yelp Dataset Challenge website https://www.yelp.com/dataset/challenge . The Yelp dataset has information on reviews, users, businesses, and business check-ins. But we focus on reviews data that includes user reviews of businesses from five different cities. We use external software FME (Feature manipulation engine) to read the json data files and convert it into csv. We only extract text reviews and star ratings and ignore the other information in the dataset for simplicity. We are focused only on the reviews that has 1 and 5-star ratings as in this project we are only finding out whether the review is bad or good. We preprocess this review’s text using bag of words approach where each unique word in a text will be represented by a number for converting the corpus into vector format. By doing this the data can be sent for training to perform the classification task by different classification algorithms.

4 External Software

We use the following libraries and external software in our project:

- **FME (Feature manipulation engine):** This external software is used to read the json data files and convert it to csv.
- **Pandas:** We use pandas library to read the csv file.
- **Seaborn and matplotlib:** We use seaborn and matplotlib libraries for exploring our dataset and visualizing it by plotting graphs.
- **NLTK:** We use nltk library for punctualtions and stopwords removal.
- **Scikit-learn:** We use scikit-learn library for multi-purpose in this project. We import countvectorizer from sklearn.feature_extraction.text for converting the text collection into a matrix of token counts. To train the dataset we import train_test_split from sklearn.model_selection. We import different classifiers to build models. For Multinomial Naïve Bayes we imported MultinomialNB from sklearn.naive_bayes; for KNN we import KNeighborsClassifier from sklearn.neighbors. To predict the accuracy, we import metrics from sklearn. To fetch f1-score, classification report and confusion matrix we import f1_score, classification_report, confusion_matrix from sklearn.metrics.

5 System Implementation

The main functional components of our projects are:

- Importing the dataset
- Text pre-processing
- Vectorization
- Training and Testing the data
- Comparison of accuracies of three different classifiers

5.1 Importing dataset

The data set is downloaded from the yelp dataset challenge website. It is in json format and is converted to csv format using FME (Feature manipulation engine) software. The csv form of data is read by using pandas.

5.2 Text pre-processing

The main issue with our data is that it is all in plain-text format. The classification algorithms will need some sort of feature vector to perform the classification task. The simplest way to convert a corpus to a vector format is the bag-of-words approach, where each unique word in a text will be represented by one number. First, we write a function that splits a message into its individual words and return a list. We also remove the common words (such as “the”, “a”, “an”, etc.), also known as stopwords. To do this, we use NLTK library. The function removes punctuation, stopwords, and returns a list of the remaining words, or tokens.

5.3 Vectorization

Now, we have our reviews as lists of tokens (also known as lemmas). To enable Scikit-learn algorithms to work on our text, we need to convert each review into a vector. We can use Scikit-learn’s CountVectorizer to convert the text collection into a matrix of token counts. We import CountVectorizer and fit an instance to our review text.

5.4 Training and testing the data

As we have finished processing the review text, it’s time to split our data into a training and a test set using train_test_split from Scikit-learn. We will use 30% of the dataset for testing.
We train our model using three different classifier algorithms:

- Multinomial naïve Bayes
- K-Nearest Neighbor
- SVM

5.41 Multinomial Naive Bayes

Multinomial Naive Bayes is a specialized version of Naive Bayes that is designed more for text documents. Whereas simple naive Bayes would model a document as the presence and absence of words, Multinomial Naive Bayes explicitly models the word counts and adjusts the underlying calculations to deal with in.
With a multinomial event model, samples (feature vectors) represent the frequencies with which certain events have been generated by a multinomial $(p_1,\ldots,p_n)$ where p_i is the probability that event i occurs (or K such multinomials in the multiclass case). A feature vector $x = (x_1,\ldots x_n)$ is then a histogram, with x_i counting the number of times event i was observed in a particular instance. This is the event model typically used for document classification, with events representing the occurrence of a word in a single document (see bag of words assumption). The likelihood of observing a histogram x is given by

$$p(\mathbf{x} \mid C_k) = \frac{(\sum_i x_i)!}{\prod_i x_i!} \prod_i p_{ki}{}^{x_i}$$

We build the Multinomial Naïve Bayes model using scikit-learn library and fit it to our training set.

5.42 K-Nearest Neighbor

K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). KNN has been used in statistical estimation and pattern recognition already in the beginning of 1970's as a non-parametric technique.
A case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its K nearest neighbors measured by a distance function. If K = 1, then the case is simply assigned to the class of its nearest neighbor. Choosing the optimal value for K is best done by first inspecting the data. In general, a large K value is more precise as it reduces the overall noise but there is no guarantee. Cross-validation is another way to retrospectively determine a good K value by using an independent dataset to validate the K value. Historically, the optimal K for most datasets has been between 3-10. That produces much better results than 1NN.
We build the KNN model using scikit-learn library and fit it to our training set.

5.43 SVM

A Support Vector Machine (SVM) is a supervised machine learning algorithm and discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (*supervised learning*), the algorithm outputs an optimal hyperplane which categorizes new examples. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have; for this project we have n = 2 as star 1 and star 5 ratings) with the value of each feature being the value of a coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well (look at the below snapshot).
The operation of the SVM algorithm is based on finding the hyperplane that gives the largest minimum distance to the training examples. Twice, this distance receives the important name of margin within SVM's theory. Therefore, the optimal separating hyperplane *maximizes* the margin of the training data. In the SVM problem, Lagrangian formulation for finding the largest minimum distance is:

$$\min L_P = \tfrac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} a_i y_i \left(\mathbf{x}_i \cdot \mathbf{w} + b\right) + \sum_{i=1}^{l} a_i$$

where l is the number of training points.

5.44 Testing the model

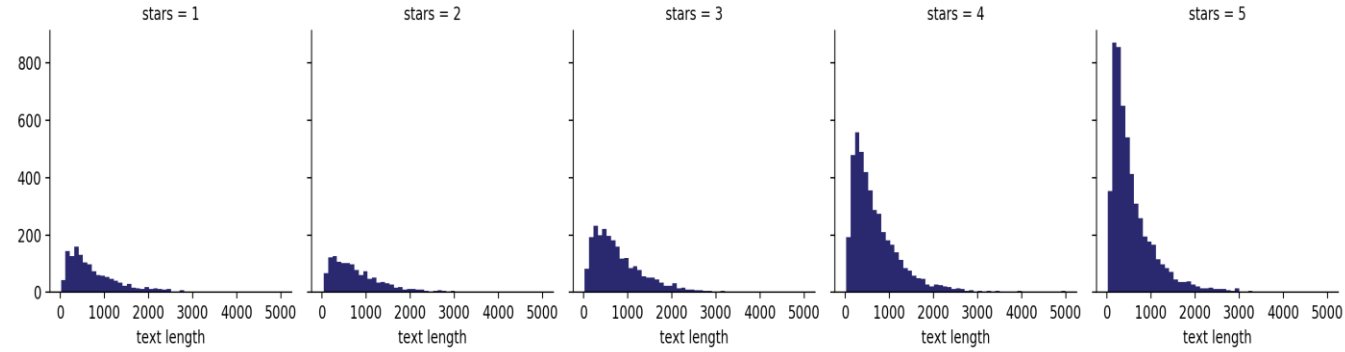
Our model has now been trained. It's time to see how well it predicts the ratings of previously unseen reviews. First, we store the prediction in separate dataframe and then evaluate our predictions against the actual ratings using confusion_matrix and classification_report from Scikit-learn.

5.5 Comparing the accuracies

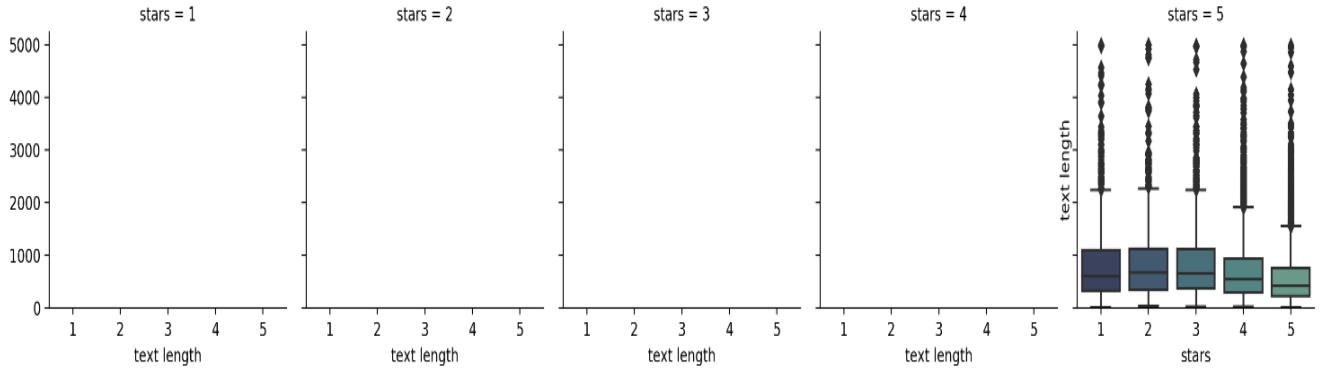
After testing our model, we finally compare the accuracies of all three different classifier algorithms.

6 Experiments and Results

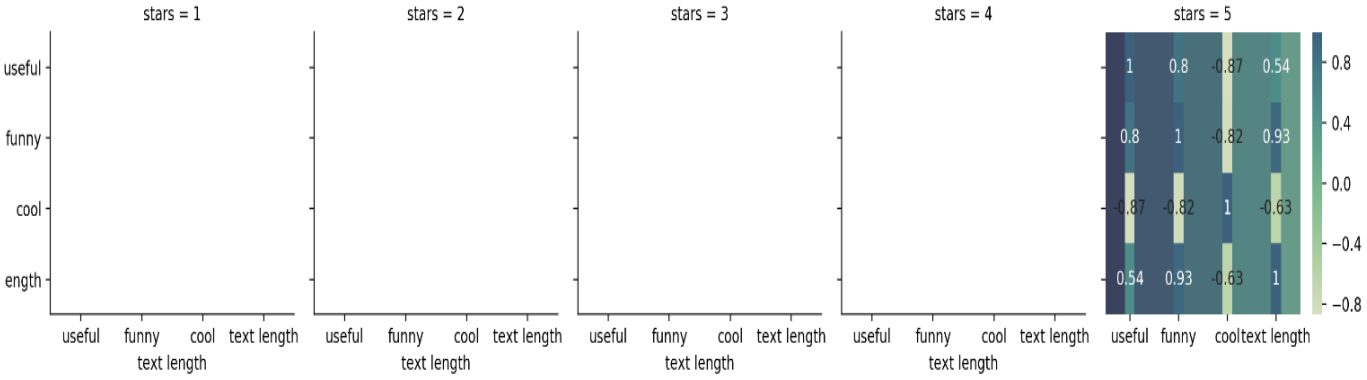
Once we import the dataset and read the csv file using pandas, we tried exploring our dataset and visualizing it by using seaborn and matplotlib library. Results are shown below.



Histograms of text length distributions for each star rating. Notice that there is a high number of 4-star and 5-star reviews.



Box plot of text length against star ratings.



Heat map of correlations between cool, useful, funny, and text length.

After training the models, we generated the classification report and confusion matrix while we tested the data.
We use Precision and Recall as the evaluation metric to measure our rating prediction performance. Our Oracle is the metadata star rating. We compare our prediction with the metadata star rating to determine the correctness of our prediction. Precision and Recall are calculated respectively by the equations below:

$$Precision = \frac{tp}{tp + fp}$$

$$Recall = \frac{tp}{tp + fn}$$

where tp, fp, fn are the number of True Positives, False Positives, and False Negatives respectively.

Below are the classification reports for the three classifiers:

Multinomial Naïve Bayes:

Star ratings	Precision	Recall	F1-Score	Support
1	0.81	0.76	0.78	423
5	0.94	0.95	0.95	1644
Avg/Total	0.91	0.91	0.91	2067

K-Nearest neighbor:

Star ratings	Precision	Recall	F1-Score	Support
1	0.52	0.40	0.45	423
5	0.86	0.90	0.88	1644
Avg/Total	0.79	0.80	0.79	2067

SVM:

Star ratings	Precision	Recall	F1-Score	Support
1	0.00	0.00	0.00	423
5	0.80	1.00	0.89	1644
Avg/Total	0.63	0.80	0.70	2067

The accuracies were obtained after training and testing the model.
The accuracies obtained are:

- Multinomial Naïve Bayes: 91.38 %
- K-Nearest Neighbor: 80.16 %
- SVM: 79.53 %

7 Conclusions

In conclusion, we have experimented with various supervised learning algorithms to predict star ratings of the Yelp dataset using review text alone. We evaluated the effectiveness of different algorithms based on precision and recall measures. We compared the accuracies of all three algorithms. Thus, we conclude that Multinomial Naive Bayes combined with stop words and stemming is the best in our context of sentiment analysis. Possible improvement could be improving the accuracy by trying with few other algorithms.

8 References

[1] X. Yun, X. Wu and Q. Wang, "Sentiment Analysis of Yelp's Ratings Based on Text Reviews", pp. 1-5, 2014.
[Online]. Available: http://cs229.stanford.edu/projects2014.html

[2] Learning Sentence Vector Representations to Summarize Yelp Reviews. [Online]. Available: http://cs224d.stanford.edu/reports/KhoslaNeal.pdf

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

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

[5] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? sentiment classification using machine learning techniques," Proceedings of the ACL-02 conference on Empirical methods in NLP, 2002.

[6] K. Dave, S. Lawrence, and D. Pennock, "Mining the peanut gallery: Opinion extraction and Semantic classification of product reviews," Proceedings of the 12th international conference on World Wide Web, pp. 519–528, 2003.

[7] M. Hu and B. Liu, "Mining and summarizing customer reviews," Proceedings of the 10th ACM SIGKDD international conference on Knowledge discovery and data mining, 2004.

[8] [Online]. Available: https://www.yelp.com/dataset_challenge/dataset