Yelp Dataset Review Rating Prediction Analysis

Shihao Lin
Pitzer College
1050 N. Mills. Ave, CA 91711
shilin@pitzer.students.edu

Ziyuan Shang
Scripps College
Claremont,CA,91711
zshang9431@scrippscollege.edu

Abstract

Rating prediction is one of the most important application of machine learning. In this project, we explore the Yelp dataset and are looking for appropriate machine learning algorithms to predict the star users will assign to a business through the content of review they wrote. We use the basic topic modeling processed the data and use classification algorithms like Naive Bayes(NB) algorithm, MultinomialNB, BernoulliNB, Logistic Regression and Linear Support Vector Machine to do binary classification (positive and negative) on the dataset. Using Tf-idf vectorizer and bigram model, our highest accuracy reached 98.16%.

1 Introduction

Yelp is an American multinational corporation headquartered in San Francisco, California. It was founded in 2004, and dedicated to helping people find a great local business like restaurants and hair salons. The company allows users to give star ratings and write reviews for the business. More than 155 millions of reviews have been written by Yelp users by the end of Q1 2018. Among these reviews, myriads of hidden information can be extracted, and this information can be beneficial to business owners as well as customers. By studying these reviews, owners can be more aware of things that they need to improve, and customers will be able to find better matches for their own tastes. However, how to extract the most important information has remained a big challenge for a long time. The project specifically focuses on building a model that can extract useful features from millions of reviews. Processing these reviews using sentiment analysis, finding the most featured words and predicting the proper ratings solely based on review text.

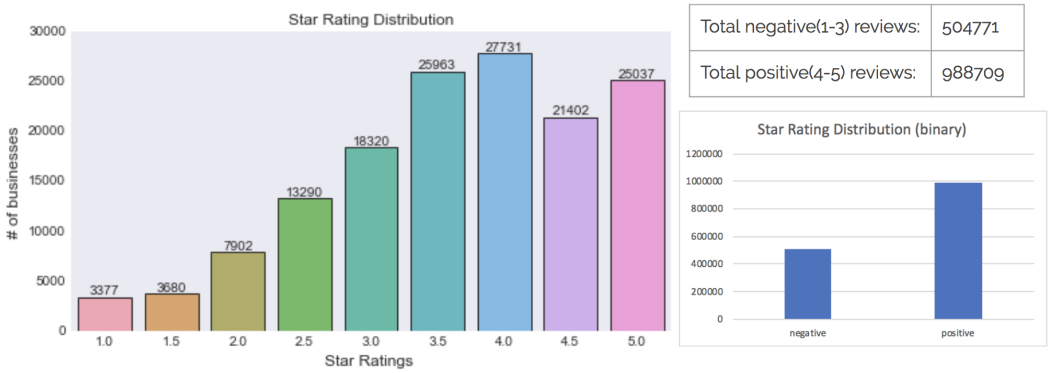
2 Data description

Yelp's datasets include six files, which are business.json, review.json, user.json, checkin.json, tip.json, and photos. In our midterm modeling, we used the business.json and review. json. Later, we will further explore the review.json for our final modeling. In the following subsections, we are going to illustrate the datasets and its format.

In this dataset, the total organizations are 146702, the total users are 1326101, the total reviews are 1493480, the total negative reviews (from rating scale 0 to 4) are 504771, and total positive reviews (from rating scale 4 to 5) are 988709.

The above figure 1 illustrates the review star distribution. Based on figure 1, we can notice that the total start reviews are mostly made up by positive reviews.

Submitted to 32nd Conference on Neural Information Processing Systems (NIPS 2018). Do not distribute.



Total negative(1-3) reviews:	504771
Total positive(4-5) reviews:	988709

Figure 1: Star Rating Distribution

2.1 Files Description

The file Business.json contains business data including location data, attributes, and categories. More specific, this file includes information of city, state, latitude, and longitude of the store, stars, review counts, business parking, ambience, price level, Good for meal and etc. In the later section, we would set up the model to illustrate the correlations between all of those attributes and business rating. In other words, we are going to build up a model to predict the business rating applying some of the features to different types of existing algorithms to train the computer.

The file Review.json contains full review text data including the user_id that wrote the review and the business_id the review is written for. We will not use this for the midterm report. But we will use this file for the final modeling to predict a solo review star, which is based on its review text.

3 Data Analysis

Before we actually start analyzing the data, we transform the format from json to csv, which is convenient for us to analyze the data. In the following subsections, we generate various graphs to help visualize the relationship between different feature.

3.1 Location v.s Review counts

In the first part, we link the review counts to states. We find out that Arizona has the most reviews, which is a bit surprising. The topmost reviews states are Arizona, Nevada, Oregon, North Carolina, and Ohio. Surprisingly, we find that California is not on this list. We also link the review counts to the city. We find that Las Vegas has the most reviews, which is 22113 and is also the main component of the total reviews of Nevada. The following figure 2 illustrates the topmost reviews city in the U.S. By using the latitudes and longitudes of business organizations and its review counts, we can plot out a graph about that information, just like the following figure 3 about Las Vegas and Phoenix. Each review will represent a small black dot. The more reviews in the same location, the dot will be darker. Therefore, we can use this graph to predict where is the most popular place in a city.

3.2 Review Count v.s. rating

In figure 4 "Review count v.s. Rating", we label the number of review in one single business organization into horizontal axis, and the probability to get a positive review into vertical axis. We found that if a business has fewer reviews, the probability of it to be rated as positive is around 50%. The business organization with a medium amount of review tends to have high variance on its probability to be rated as positive. The organization with large reviews often tend to have the probability to be rated as positive.

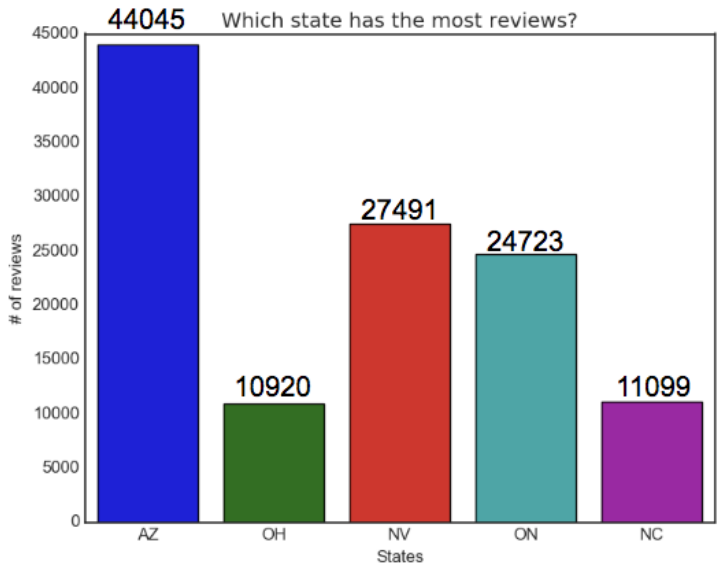


Figure 2: What state has the most reviews?

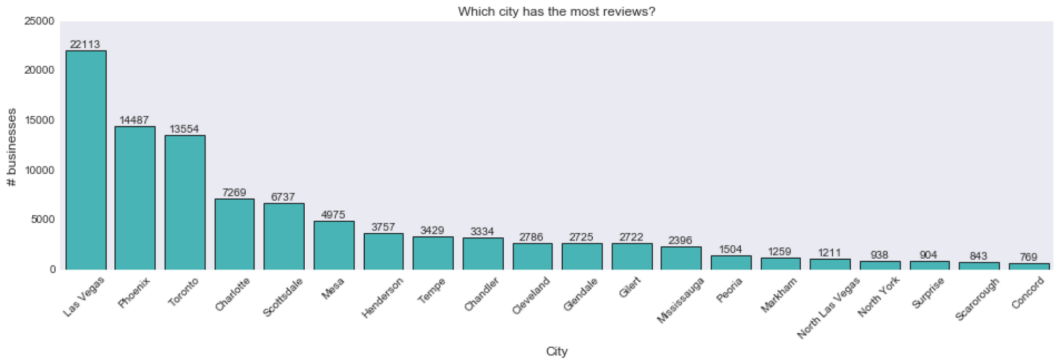


Figure 3: Which city has the most reviews

3.3 Reservation v.s. Rating

We want to discover the relationship between whether an organization accepts reservation would influence it to be rated as positive. Therefore, we analyzed the data and put it into a histogram as presented in figure 5. On the horizontal axis, 0 represents that reservation service is unavailable, and 1 represents that reservation service is available. The vertical axis shows the probability for each type of organization to be rated as positive. As a result, we found the probability of being rate as positive for both types are very similar and centering around 50 percentage. Thus, we conclude that there is no such correlation between the service of reservation and business rating.

3.4 Price v.s. Rating

The price level of a restaurant is always a big factor to influence people's decision of selecting a restaurant. Sometime, people would think an expensive restaurant tend to taste better than a cheap restaurant. In this paper, we are going to present our finding of the correlation between price level and business rating. By plugging the price level data and business rating data into "Seaborn" analyzer, we get a histogram in figure 6. Each number on the horizontal axis represents different price levels. "1.0" means very cheap, and "4.0" means very expensive. The vertical axis shows the probability for different price levels of an organization to be rated as positive. As a result, we found the probability of all four different price level are very similar and close to 50 percentage, and the probability of price

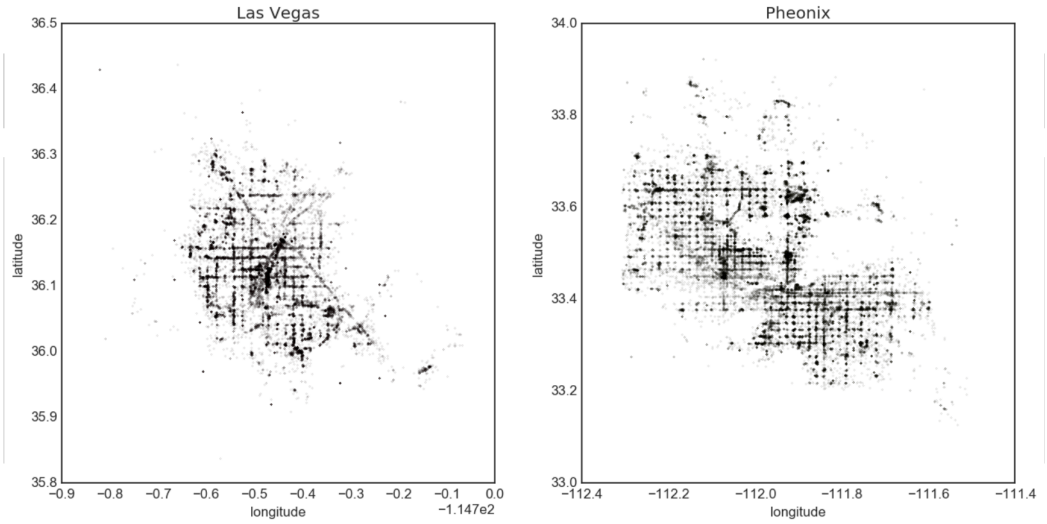


Figure 4: Business popularity on city map

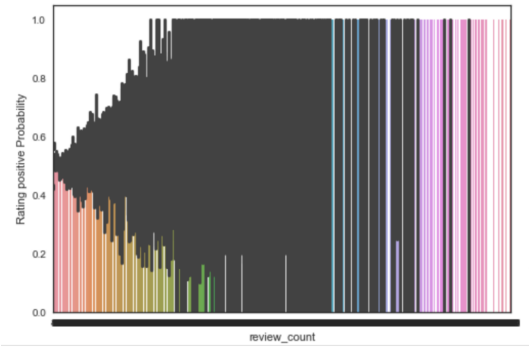


Figure 5: Review Count v.s. Rating

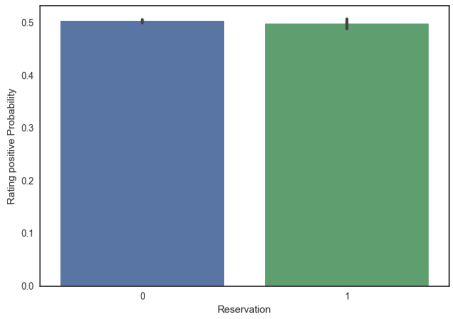


Figure 6: Reservation v.s Rating

level 4 is slightly lower than the price level. Thus, the result indicates the price level may not have much influence on Business Rating. A very expensive restaurant may not always taste very good.

3.5 Ambience v.s. Rating

Different type ambience of a restaurant tends to have different chance to be rated as a positive good restaurant. We want to discover the correlation between ambience and rating. In the yelp dataset, it provides 10 categories for ambience, "None", "casual", "divey","intimate","romantic", "trendy","classy",'touristy", "hipster", and "upscale". Then we plugged all of those data with its business rating into "Seaborn" analyzer. We get a bar diagram in figure 7. Different categories are labeled in the horizontal axis. The vertical axis shows the probability for different categories of an organization to be rated as positive. As we observed, "upscale" ambience restaurant tend to have higher chance to be rated as positive. However, in general, since the positive rating probability for all categories is all laid between 40 percentage and 60 percentage, the ambience does not seem to have much correlation with the business rating.

3.6 City v.s. Rating

We are curious about whether the overall quality of food in some city is higher than other cities. Therefore, we input all organizations' city location and corresponding business rating into "Seaborn" analyzer and get a histogram as shown in figure 8. The cities' names are labeled in a horizontal axis, and the corresponding positive rating probability is presented by a vertical axis. The different color

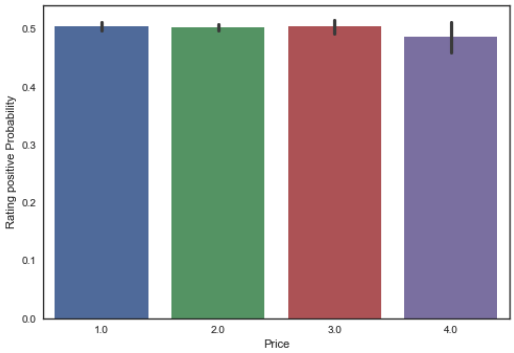


Figure 7: Price v.s. Rating

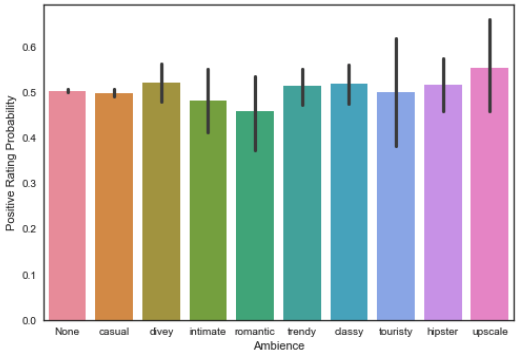


Figure 8: Ambience v.s Rating

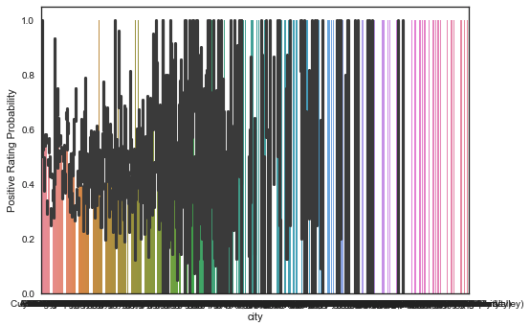


Figure 9: City v.s. Rating

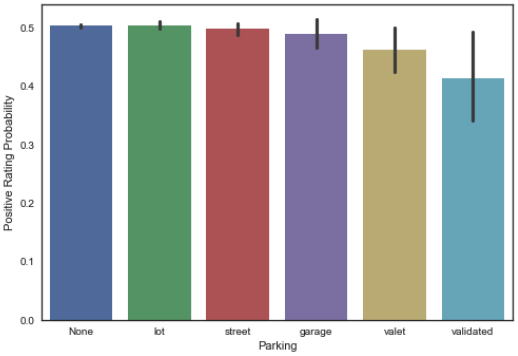


Figure 10: Parking v.s Rating

bar represents the different city, and the black bar is the variance of the probability. By observation, we found some city definitely have higher overall quality restaurant than others.

3.7 Parking v.s. Rating

The transportation in most of the place in the U.S. is driving. In average, every household has around 2 cars in the U.S. Thus, we think whether a restaurant would provide parking service is essential to its rating. Then we plug the data about its restaurant parking situation and corresponding business rating into "Seaborn" analyzer and get a histogram. As in figure 9, we observe that the positive rating probability for all categories is under 50 percent. Thus, the service of parking does not seem to have much influence on the business rating.

3.8 Good For Meal v.s. Rating

At first, we consider that different meal type may have an influence on the business rating. For example, restaurant, which focuses on making breakfast, in general, may have higher business rating than those focuses on making brunch. In yelp data, it provides 7 categories for "Good For Meal", which are "None", "late night", "dessert", "breakfast", "dinner", "lunch", and "brunch". While we plugged all of those data and corresponding business rating, we get a bar diagram, as shown in figure 10. The meal categories are labeled in a horizontal axis, and the corresponding positive rating probability is presented by a vertical axis. By observation, we found that the positive rating probability for all categories are very similar and around 50 percentage. Thus, distinguishing meal specialty of a restaurant may not help to distinguish its business rating.

4 Business Analysis

Before actually applying the data into the algorithm, we decide to filter out some features of the dataset. In the end, we remain the three features, review_count Reservation, and Parking, to predict

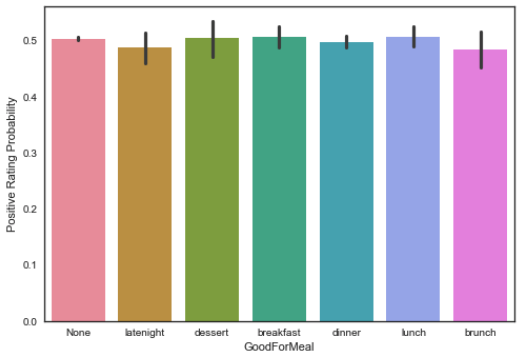


Figure 11: Good For Meal v.s Rating

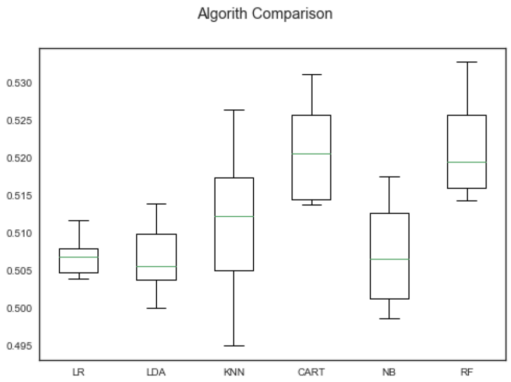


Figure 12: Algorithm.png

the business rating. We set 75 percentage dataset as the training dataset, and remain 25 percentage dataset as the testing dataset. Then we purposely select six different methods to predict the outcome, which is "Logistic Regression", "Linear Discriminant Analysis", "K-Neighbors Classifier", "Decision Tree Classifier", "Gaussian Naive Bayes", and "Random Forest Classifier" from $sklearn$ package. All of six methods could be used to analyze a binary outcome.

In the previous section, although the features in the dataset are independence to each other, we notice that most of the features in this dataset do not have much correlation with the business rating. Therefore, we assume that the accuracy for those algorithms to have a right prediction is low. The overall result is present in the following table, and the visualized version is in figure 12.

By observation, the overall accuracy is very low and around 50 percentage. We observed that " Logistic regression" method has the lowest variance, "K-nearest Neighbors classifier" has the highest variance, and "Random Forest classifier" brings up the highest accuracy, which is yielding roughly 52.11%. Although "Decision tree classifier" is the underlying concept for "Random Forest classifier", the accuracy performance does not have much different. In conclusion, the correlation between the feature and its outcome is essential for generating a high accuracy prediction.

Algorithm	Accuracy Percentage
Logistic_Regression_classifier	0.506369 (0.003203)
Linear_Discriminant_Analysis	0.506685 (0.004129)
K-Neighbors_classifier	0.511014 (0.009426)
Decision_Tree_classifier	0.520832 (0.006517)
Gaussian_Naive_Bayes	0.506978 (0.006656)
Random_Forest_classifier	0.521114 (0.006188)

5 Review Analysis

In our review analysis, we mainly explored the relationship between review content (what words does user use) and rating of the corresponding business. We first attempted our reviews analysis on a smaller sample. Form our reviews dataset, we generated a random sample of 3000 reviews. We will scale the code to perform analysis for the entire dataset in the future.

5.1 Data preparation

The data used in this analysis comes from the preprocessed review.csv mentioned in Section 3. The only useful columns are *text* - the actual review content given by the user in a string format, and *review_stars* - the star rating associated with the review. A third column named *Rating* is created based on *review_stars* - reviews with 1, 2 stars are considered negative; reviews with 4 or 5 stars are considered positive.

5.2 Feature selection

Prior to fitting the model and using machine learning algorithms for training, we need to represent the text document as a feature vector. A commonly used model in Natural Language Processing is the *bag of words* model. It first comes the creation of the vocabulary — the collection of all different words that occur in the training set and each word is associated with a count of how it occurs. The vocabulary can then be used to construct the d-dimensional feature vectors for the individual documents where the dimensionality is equal to the number of different words in the vocabulary. Before we do the vectorization, we need to pre-processing the code.

5.2.1 Text pre-processing

First, we performed several necessary pre-processing on review test, including tokenizing, removing stop words and stemming.

Tokenizing refers to the act of separating a sentence into individual word tokens. In our analysis, we used the default *word_tokenizer* in the *nltk* package under Python. As an example, if we pass the following sentence:"The python programmer named pythoner is pythoning a game pythonly." into the *word_tokenizer*, the returned result will be

$$'The',' python',' programmer',' named',' pythoner',' is',' pythoning',' a',' game',' pythonly'$$

After tokenization, we need to remove the common stop words in English. Stop words are words that appear extremely common but do not convey much meaning or sentiment of bodies of text. We used the bad of stop words gathered by the *nltk* package and eliminated the words that falls into this bag from each piece of text. Using the above example, after removing the stop words, the returned result is

$$'The',' python',' programmer',' named',' pythoner',' pythoning',' game',' pythonly'$$

Observed that the words 'is' and 'a' are removed from the list, whereas the words that conveying meanings are kept.

Then we stemmed the individual words in each piece of text using Porter Stemmer(C.J. van Rijsbergen and Porter,1980). Stemming refers to "the process of reducing inflected (or sometimes derived) words to their word stem, base or root from generally a written word form". In particular, Porter Stemmer is the de facto algorithm developed in 1980, and it was widely used to stem English words. In our analysis, we used the *nltk* implementation of Porter Stemmer to bring words from text to its stem form. Here, we assumed that words with the same stem convey roughly the same meaning, and therefore are weighted equally in sentiment analysis.

After passing the above list to a Porter Stemmer, the list returned is in the following form:

$$'the',' python',' programm',' name',' python',' python',' game',' pythonli'$$

Observed that all python-related words, except 'pythonly', are being turned into its stemmed for 'python'. The single exception of the adverb might be an example of under stemming.

5.2.2 Spare Matrix Representation

The final step towards model training is transforming the body of text into a spare matrix representation. We have explored two different ways: Count Vectorizer and Tf-idf Vectorizer.

The idea behind Count Vectorizer is quite simple: we first identify all the unique words that are present in all of the review, then create a matrix whose columns are individual words and rows are individual documents. The values represent the number of times the word appeared. Let D_1 and D_2 be two documents in a training dataset: D_1:"Each state has its own laws."; D_2:"Each country has its own culture."

Based on these two documents, the vocabulary could be written as

$$V = \{each : 1, state : 1, has : 2, its : 2, own : 2, laws : 1, every : 1, country : 1, culture : 1\}.$$

Then, the bag of words representation of these two sample documents will be

	each	state	has	its	own	laws	every	country	culture
x_{D_1}	1	1	1	1	1	1	0	0	0
x_{D_2}	0	0	1	1	1	0	1	1	1
$\sum$	1	1	2	2	2	1	1	1	1

Then we can sent it into probabilistic models for evaluation.

The Tf-idf Vectorizer is more complicated yet more powerful transformation approach. "Tf-idf" is an acronym than stands for “Term Frequency – Inverse Document” Frequency which are the components of the resulting scores assigned to each word. "Term Frequency"summarizes how often a given word appears within a document, and "Inverse Document Frequency"downscales words that appear a lot across documents. In Tf-idf calculation, the term frequency is multiplies with the idf component:

$$Tf - idf = tf_n(t, d) \cdot idf(t)$$

Let $tf_n(t, d)$ be the normalized term frequency, and idf, the inverse document frequency, which can be calculated as follows

$$idf(t) = log(\frac{n_d}{n_d(t)})$$

where n_d is the total number of documents and $n_d(t)$ is the number of documents that contain the term t. The overall effect of Tf-idf transformation is that more weights are transferred from terms that are more ubiquitous across texts to those that are more unique to specific texts.

We also apply N-gram model on Tf-idf vectorizer. In the n-gram model, a token can be defined as a sequence of n items. Here, we found that n-gram of size 2 yielded the highest accuracy , which called bigrams. For word phrases like "never disappointed" and "not good", using bigrams can help the model understand the real meaning of the review.

5.3 Model training and evaluation

5.3.1 Training

For predictions, we have tested a few machine learning algorithms that can make binary (positive and negative) classifications using different sets of features. We have explored from the *nltk* package, trained on sparse matrix build by only individual words as features. We also used several more advanced algorithms. Including MNB, BernoulliNB, Logistic Regression, Linear SVC from the *sklearn* package.

5.3.2 Evaluation

Below is the graph comparing the accuracy of different algorithms we’ve tried. We can observe that the Naive Bayes Classifier performed poorly, yielding roughly 50.87%. Then, we used several more advanced algorithms, including MNB, BernoulliNB, Logistic Regression, Linear SVC, the test accuracy are as follows:

Classifier	3000 data points	90000 data points
MNB_classifier	78.87%	90.33%
BernoulliNB_classifier	79.67%	82.08%
LogisticRegression_classifier	86.33%	94.46%
LinearSVC_classifier	85.64%	93.37%

We first observed that larger the dataset, higher the accuracy. We see that the Logistic Regression are producing highest result with 94.96% of accuracy. The performance still can be improved. Next, we will compare between different vectorizers: Count vectorizer, Tf-idf vectorizer and Tf-idf with bigram model.

We sort and show the word features by their coefficients. From Figure 13, we can see that our model built using count vectorizer is not that ideal.We can’t observe obvious distinction from the two groups.

The Tf-idf vectorizer performed much better (Figure 14). We can clearly observe that small coefficients are negative words,and larger coefficients are positive words.

```
Smallest Coefs:
['preselected' 'negitive' 'excavated' 'phs' 'wrung' 'exodus' 'miscues'
 'montmartre' 'portishead' 'crimini' 'sprechers' 'conglomerates'
 'stinginess' 'sunridge' 'infuriated']

Biggest Coefs:
['jav' 'paddies' 'wihtout' 'dickinson' 'dobra' 'wizards' 'ssssooo' 'vt'
 'koval' 'vcr' 'bosas' 'robotics' 'preconceived' 'abusers' 'seagull']
```

Figure 13: Word features with count vectorizer

```
Smallest Coefs:
['worst' 'mediocre' 'disappointing' 'unprofessional' 'bland' 'horrible'
 'meh' 'downhill' 'tasteless' 'rude' 'terrible' 'lacked' 'awful' 'inedible'
 'poisoning' 'flavorless' 'unacceptable' 'worse' 'overpriced' 'poor'
 'disappointment' 'poorly' 'rudest' 'underwhelming' 'ok']

Biggest Coefs:
['amazing' 'delicious' 'great' 'excellent' 'awesome' 'best' 'fantastic'
 'perfect' 'perfection' 'highly' 'pleasantly' 'incredible' 'phenomenal'
 'love' 'perfectly' 'hesitate' 'notch' 'wonderful' 'thank' 'gem' 'grateful'
 'heaven' 'deliciousness' 'outstanding' 'superb']
```

Figure 14: Word features with tf-idf vectorizer

```
Biggest Coefs:
['amazing' 'great' 'delicious' 'best' 'excellent' 'awesome'
 'be disappointed' 'fantastic' 'perfect' 'not disappointed' 'love'
 'wonderful' 'incredible' 'outstanding' 'not only' 'never disappointed'
 'highly recommend' 'loved' 'you wont' 'friendly' 'perfection' 'definitely'
 'been disappointed' 'perfectly' 'even better' 'phenomenal'
 'will definitely' 'happy' 'love this' 'so good' 'five stars' 'heaven'
 'favorite' 'thank you' 'honest' 'thanks' 'no charge' 'definitely be'
 'yummy' 'good' 'superb' 'professional' 'on point' 'better than' 'thank']
```

Figure 15: Word features with tf-idf vectorizer and bigrams model

Bigrams (Figure 15) is able to capture negations like "not good" and "not recommend", also compliment like “never disappointed” and "not disappointed". The results showed that including bigrams indeed increased test accuracy (Figure 16).The highest accuracy percentage now is 98.16% using Tf-idf vectorizer with bigrams model and LinearSVC classification algorithm.

6 Future Directions

6.1 Multi-classification

In this section, instead of making a prediction of binary outcome, we try to use "Multi-Naive Bayes", "Bernoulli Naive Bayes", "Logistic Regression", "Stochastic Gradient Descent Classifier", "Support Vector Classifier", "Linear-Support vector Classifier", and "Random Forest Classifier" to predict the review rating from one star to five stars. The overall runtime for predicting five stars review takes much longer to predict the binary outcomes. At first, we start with testing a set of three thousand data points. 1500 data points were set the training set and 1500 data points were set as the testing set. Since we try to make a much accurate prediction, the overall accuracies of seven algorithms are very low, as we can observe from the figure 16. The highest accuracy is 44.40 percentage, which is produced by Logistic Regression. In order to achieve higher accuracy, a set of ten thousand data points were ran. Then five thousand data points were set up as the training set and five thousand data

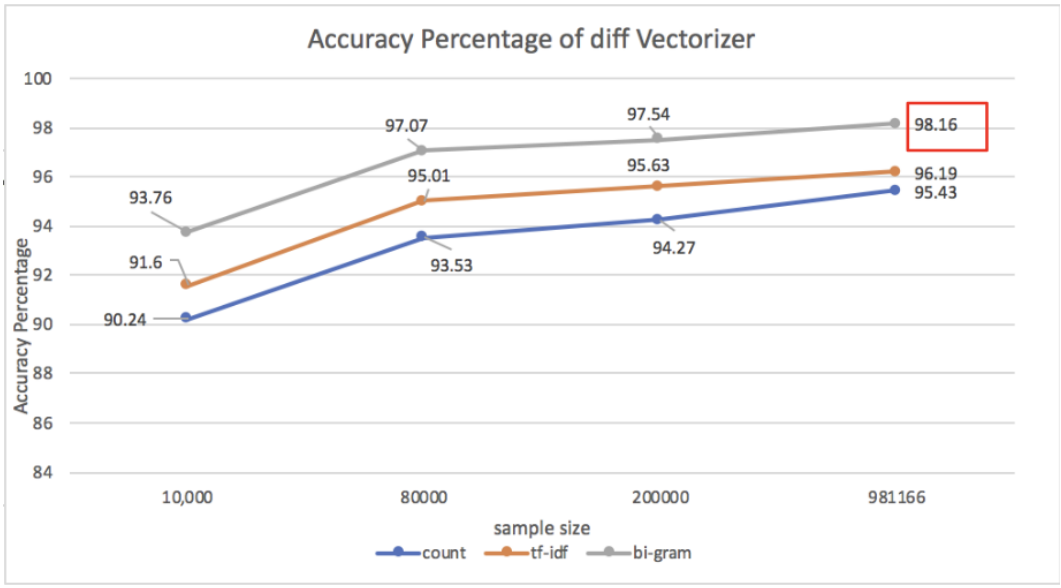


Figure 16: Accuracy percentage of different vectorizer

points were set up as the testing set. In general, the accuracy of the algorithm have been increased, except the "Support Vector Classifier." But the accuracies are still very low. At this moment, the runtime becomes much longer. At the end, fifty thousands data points are separated into a twenty-five thousand training set and a twenty-five thousand testing set. The overall accuracies do not increase much. The highest accuracy is 60.70 percentage.

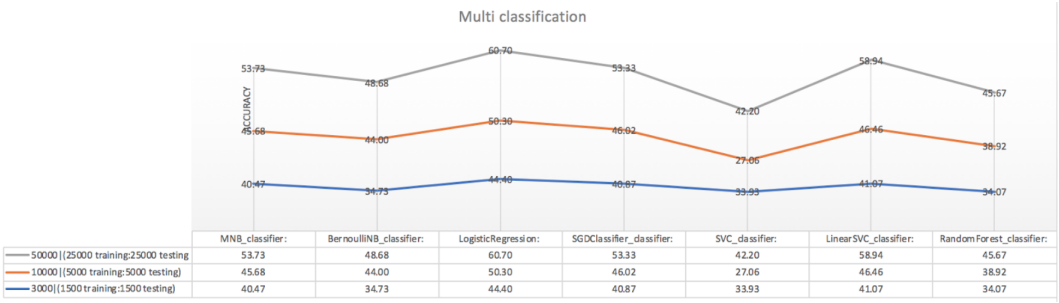


Figure 17: Accuracy percentage of multi-classification

6.2 Building an LDA model

Topic Modeling is a technique to extract the hidden topics from large volumes of text. Latent Dirichlet Allocation(LDA) is a popular algorithm for topic modeling. LDA's approach to topic modeling is it considers each document as a collection of topics in a certain proportion. And each topic as a collection of keywords, again, in a certain proportion.Here, we will try to build an LDA model to extract general topics in Yelp's review and explore whether it helps to classify the meaning of the review content. We use Python's Gensim package to do the implementation and use pyLDAvis package do the visualization of the topics.

Like the review analysis in section 5, we first prepare the dataset (section 5.1) and preprocess the documents (section 5.2.1). Since two main inputs to the LDA topic model are the dictionary and the corpus, we create the dictionary and corpus using our preprocessed data.

Before building the LDA model, another big question is that how to find the optimal number of topics k. Here, our approach is to find the optimal number of topics is to build many LDA models with different values of number of topics (k) and pick the one that gives the highest coherence value.

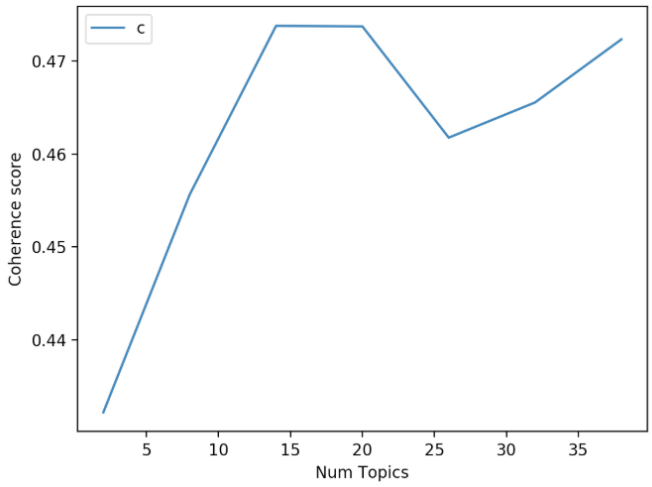


Figure 18: Topic coherence score with different number of k

Usually, we choose k that marks the end of a rapid growth of topic coherence, which offers meaningful and interpretable topics. From Figure 18, we choose k to be 20.

We visualize the topics using pyLDAvis package as follows: We can observed that LDA model not

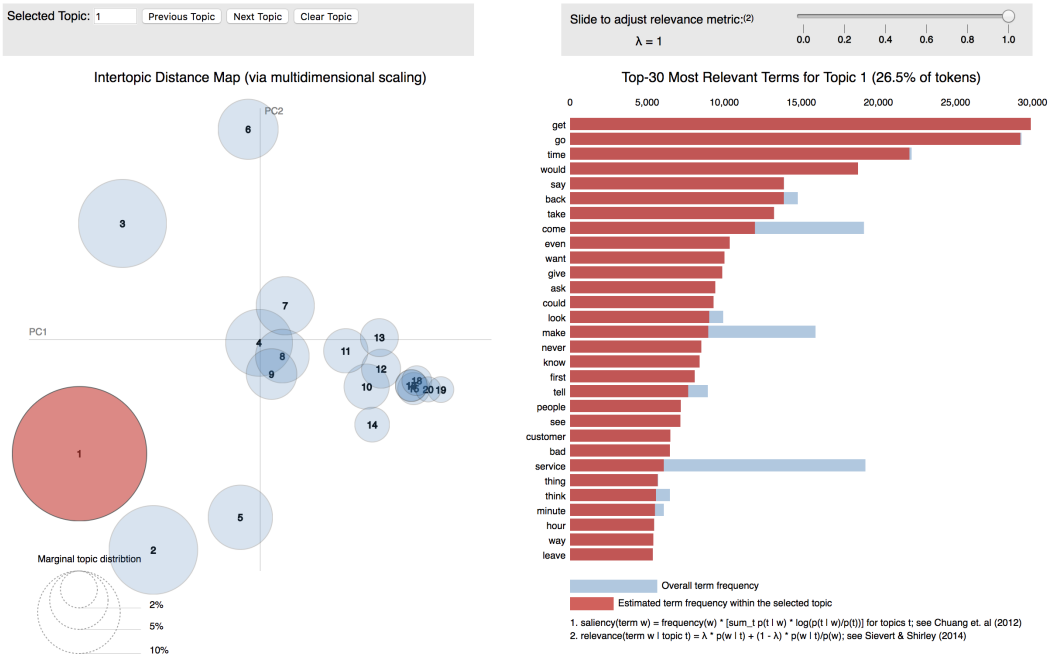


Figure 19: topics visualization

performed that well. Using the largest topic 1 as an example, the words showed the topic are mostly verbs which does not convey many meanings and does not help predict the review ratings.

7 Discussion

From Figure 16, we observe that larger the dataset, higher the accuracy. The accuracy of predicting the review rating (positive and negative) still can be improved is we use larger data points.

Recall that in our analysis, we define positive as 4 and 5 stars, negative as 1 and 2 stars. 3 stars are considered neutral and filtered in our analysis. We still need to understand the real insight of dropping the middle rank 3.

Although the LDA model (section 6.2) we built does not performed well, it can be powerful if carefully designed. The model should be further explored in future analysis.

References

[1] https://www.yelp.com/dataset-challenge.

[2] C.J. van Rijsbergen, S.R. and Porter, M. (1980). New models in probabilistic information retrieval. (British Library Research and Development Report, no.5587).