UNIVERSITY OF CALIFORNIA

Los Angeles

Yelp Review Rating Prediction:

Sentiment Analysis and the Neighborhood-Based Recommender

A thesis submitted in partial satisfaction

of the requirements for the degree

Master of Applied Statistics

by

Rui Qiao

2019

ABSTRACT OF THE THESIS

Yelp Review Rating Prediction:

Sentiment Analysis and the Neighborhood-Based Recommender

by

Rui Qiao

Master of Applied Statistics

University of California, Los Angeles, 2018

Professor Ying Nian Wu, Chair

Nowadays, the popular online review sites like Yelp have greatly affected the user purchase behaviors. Users either search for reviews to judge the quality of interested businesses or get recommendations of businesses they might like. Accordingly, this paper explores review rating predictions with two approaches. Binary sentiment analysis used vectorized review documents to predict general positive or negative attitudes in the text reviews. It gives higher prediction accuracy and adds on real-word interpretability of text reviews. A nearest neighbor collaborative filtering recommender predicts five-star ratings based on similar users and similar businesses. It predicts more comparable results to actual ratings with decent model accuracy and can make user-specific recommendations.

The thesis of Rui Qiao is approved.

Frederic R Paik Schoenberg

Hongquan Xu

Ying Nian Wu, Committee Chair

University of California, Los Angeles

2019

TABLE OF CONTENTS

LIST OF FIGURES

LISTS OF TABLES

CHAPTER 1

Introduction

1.1 Background and Motivation

Crowd-sourced online review sites like Yelp collect opinions of businesses and provide recommendations for users. Every minute 26380 reviews are posted by Yelp users. And 80% of users visit Yelp with the intention to make a purchase based on reviews. (Saleh, 2015)

The increased internet usage and smartphone penetration make online reviews a crucial role in consumer purchase decisions. The infographic posted on *InverspI* states:

90% of consumers read online reviews before visiting a business.

88% of consumers trust online reviews as much as personal recommendations.

72% of consumers will take actions only after reading a positive review.

86% of consumers will hesitate to purchase from a business that has negative online reviews.

(Saleh, 2015)

In generally, online review sites affect the purchase decisions in two ways. On the one hand, users voluntarily search for relevant reviews and form their own opinions towards the interested business based on how they perceive the sentiments from reviews. Sometimes without explicit indicators of consumers' attitudes (such as star ratings), the sentiment polarity needs to be derived solely from the textual content. On the other hand, the website recommends to users the unreviewed businesses with highest predicted ratings to develop new interests. In both cases, the prediction of ratings (either sentiment polarity or five-star ratings) is a major concern.

1.2 Problem Defining

This paper focuses on two approaches to predict review ratings. Sentiment analysis studies the textual contents of existing reviews and learns positive or negative opinions. The recommender predicts ratings based on user-business pairs and recommends to a user the unreviewed businesses with highest predicted ratings. Figure 1.1 provides an illustration of two approaches.

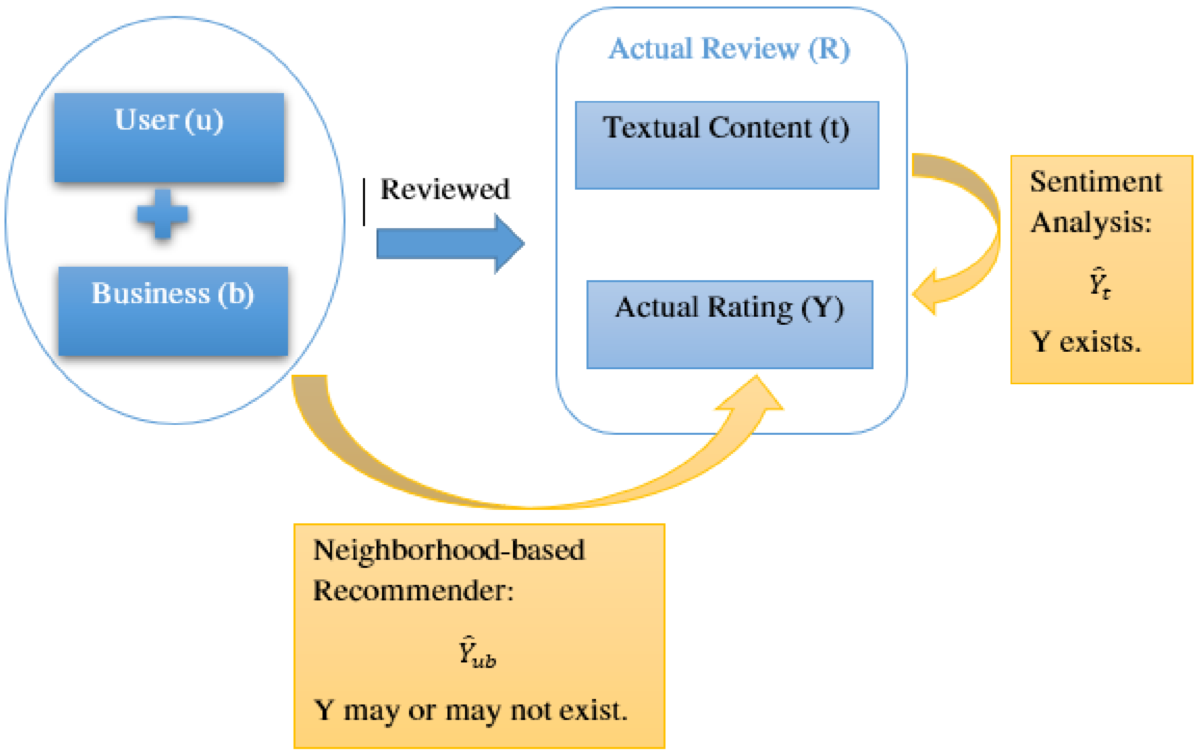


Figure 1.1 Two Approaches to Review Rating Predictions

CHAPTER 2

Data Exploration and Preprocessing

The datasets used in the paper are obtained from the Yelp Dataset Challenge Round 12. ("Yelp Dataset", 2018) There are three datasets of interest: *business*, *review*, and *user* dataset.

The steps of data preprocessing are as follows:

1) Left join *review* and *business* to get a new *review_business* data frame, which adds business information for each review record.
2) Limit *review_business* to contain only reviews for restaurants in Las Vegas.
3) Look at the time frame in review dataset. Limit *review_business* to reviews within the past 3 years that contains the majority of data and are not outdated. (See Figure 2.1) Time starts from 7/1/2015 to 7/2/2018.
4) Left Join *review_business* and *user* to get *review_business_user*. Recompute the new review counts for each user and restaurant.
5) Discard review records in *review_business_user* with review count per user < 25, review count per restaurant < 250. Recompute the review counts and average star rating of each user and business.

The resulting data frame 32637 reviews of 625 Las Vegas restaurants by 1453 Yelp users in a three-year period.

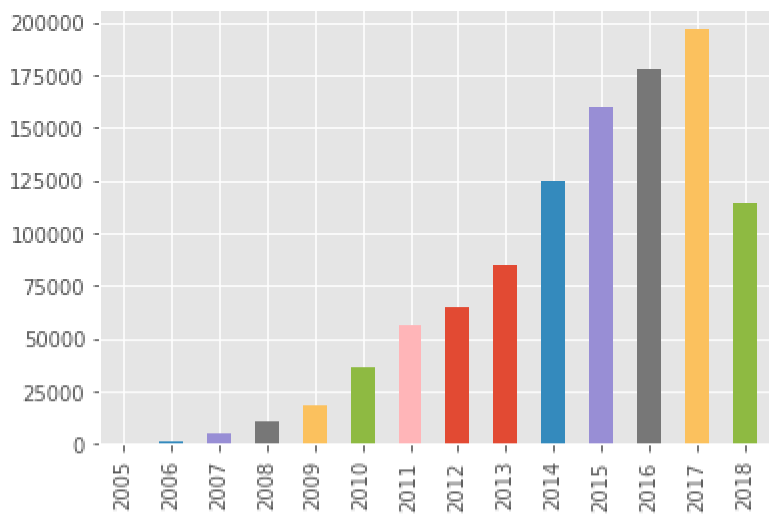


Figure 2.1 Reviews by Year

Star rating (Y) is the main thing to predict in this analysis. Nearly ¾ of the actual review ratings are either four or five out of five stars. (Figure 2.2) The multi-point rating scale will be transformed later in the per-review sentiment analysis.

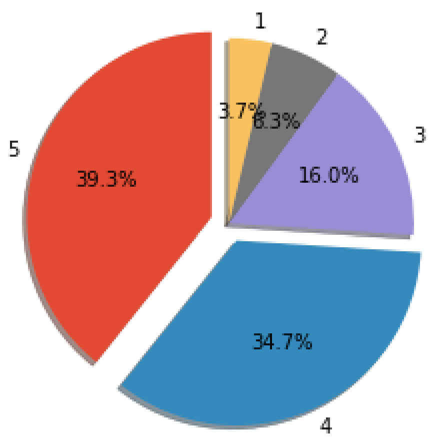


Figure 2.2 Pie Chart of Star Ratings

A further look at the rating information is on the per-user or per-business basis. After data trimming, the recomputed 4 values disclose important features for both user and business (item) and can be used to build a user-business based recommender. (See Figure 2.3)

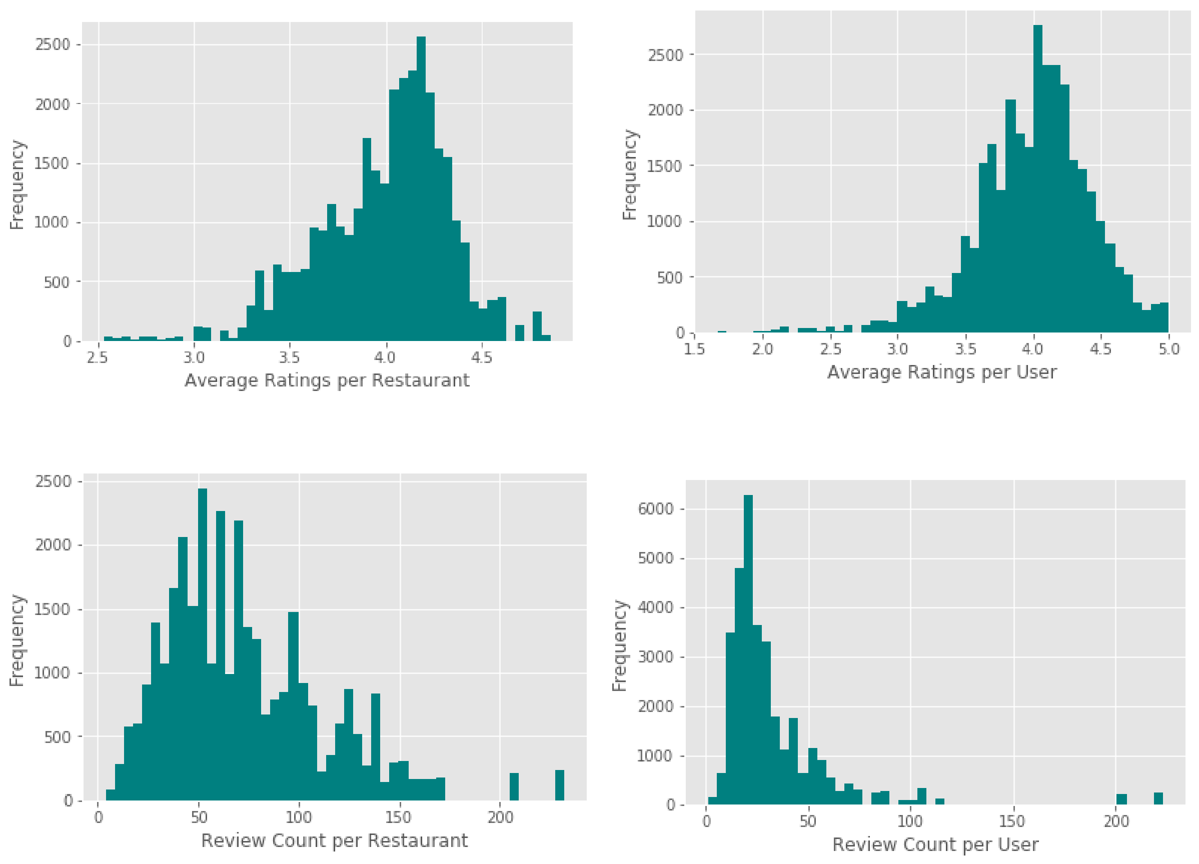


Figure 2.3 Average Rating/ Review Count per Restaurant/ User

CHAPTER 3

Methodology

This section defines and compares different classification algorithms, similarity measures and evaluation metrics. It establishes the basis of different choices in further analysis.

3.1 Binary Classification Algorithms

Table 3.1 summarizes the binary classification algorithms used for sentiment analysis in this paper based on an article. (Garg, 2018)

Algorithm	Description
Logistic Regression	Uses the logistic function to model the probability of occurrence for each categorical response. Understands the influence of several independent variables on a single outcome variable.
Navie-Bayes	Uses Bayes' theorem with the assumption of independence between every pair of features. Requires small training data and is fast. Works well in document classification.

Random Forest	Fits decision trees on various sub-samples of datasets. Uses averages to improve the accuracy and controls over-fitting. Is slow and complex to implement.
Support Vector Machine	Represents the training data as points in space separated into categories by a clear gap that is as wide as possible. Predicts the category of new examples by mapped into that space look on which side of the gap they fall. Is effective in high dimensional spaces and memory-efficient.

Table 3.1 Summary of Binary Classification Algorithms

3.2 Cosine Similarity vs. Pearson Similarity

Cosine similarity and Pearson correlation coefficient are two common similarity measures of user-item similarity.

Cosine similarity of two vectors is defined as below:

$$cosine\ similarity = \cos\theta = \frac{\mathbf{A}\cdot\mathbf{B}}{||\mathbf{A}||||\mathbf{B}||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Pearson correlation coefficient is basically the cosine similarity of the centered vectors:

$$pearson\ correlation\ coefficient = \frac{\sum_{i=1}^{n}(A_i - \overline{A})(B_i - \overline{B})}{\sqrt{(\sum_{i=1}^{n}(A_i - \overline{A})^2)(\sum_{i=1}^{n}(B_i - \overline{B})^2)}}$$

When calculating the user-item similarity, Pearson correlation often gives more accurate results and is more efficient compared to the cosine measure. Cosine similarity loses information due to reduction model and is expensive. But when data is sparse, cosine similarity outperforms Pearson similarity with higher scalability. (Dharaneeshwaran, Nithya, Srinivasan & Senthilkumar, 2017) It is because that cosine similarity treats non-existing ratings as 0s, while Pearson correlation normally drops them.

The similar review search engine in Section 4.3.3 uses cosine similarity since with feature extraction the collection of reviews is encoded into a sparse matrix of word frequencies. Pearson similarity is used in building a recommendation system for accuracy and efficiency since the user-item matrix is non-sparse.

3.3 Evaluation Metrics

The recommendation system in this paper is training and testing on the whole dataset. Commonly used accuracy measure of root mean squared error (RMSE) and accuracy are used to provide some hints. (See Section 5.3)

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{j-1}^{n}(y_j - \hat{y}_j)^2}$$

Set aside the normal metrics like root mean squared error (RMSE), metrics specific to binary classification should be considered. Most evaluation metrics for binary classification can be defined by four parameters. A confusion matrix for binary classification that defines those parameters are shown in Table 3.2.

	Predicted 0	Predicted 1
Actual 0	True Negative (TN)	False Positive (FP)
Actual 1	False Negative (FN)	True Positive (TP)

Table 3.2 Confusion Matrix for Binary Classification

Five commonly used metrics are calculated from the confusion matrix. (See Table 3.3)

Metrics	Formula
True Positive Rate (TPR), or Recall, Sensitivity	$TPR = \frac{TP}{FN + TP}$
False Positive Rate (FPR)	$FPR = \frac{FP}{FP + TN}$
Positive predictive Value (PPV), or Precision	$PPV = \frac{TP}{TP + FP}$
Accuracy (Acc)	$ACC = \frac{TP + TN}{TP + FP + FN + TN}$
F1 Score (F1)	$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precison + Recall}$ $= \frac{2}{\frac{1}{Precison} + \frac{1}{Recall}}$

Table 3.3 Evaluation Metrics Derived from Confusion Matrix

Classification accuracy is the most intuitive evaluation metric with good interpretability. However, accuracy paradox may occur with unsymmetrical data. If the instance of one category is dominant in the dataset, the high accuracy of the predicting model may not be very useful. (Afonja, 2017)

F1 Score conveys the balance between precision and recall score. Precision tells, among all the instances that the model classified into the positive class, the percentage of instances that are actual positive class. Recall is the percentage of the actual positive class that can be classified as positive. For both metrics, high scores are desired, with contributes to a high F1 score. Compared to model accuracy, F1 score is less interpretable but suits better for uneven class distribution. (Garg, 2018)

A ROC (Receiver Operating Characteristics) curve is the probability curve plotted with True Positive Rate (or recall) against False Positive Rate. False positive rate tells the percentage of actual negative class instances that are incorrectly classified in positive predicted class. AUC (Area Under the Curve) represents the degree of separability. ROC and AUC are also good for imbalanced data. They suit best in cases when a class of smaller number is important. (Swalin, 2018)

CHAPTER 4

Sentiment Analysis

Natural Language Processing (NLP) helps computers to get a human-level understanding, interpretation and manipulation of language. To make purchase decisions, a Yelp user often read others' reviews of a business to know whether the business is rated negatively or positively. Sentiment analysis simulates this process and let machine auto-detect the sentiment polarity from the text review alone. Besides the prediction models, the top negative or positive words and reviews with similar sentiments can also help understand the review reading of users.

4.1 Data Implementation

The train/ test split ratio is set to 70/ 30, which creates 22845 rows of training data and 9792 rows of testing data. Cross-validation is used in some models with relative low runtime for comparison.

4.1.1 Star rating conversion

The specific five-star rating can be difficult to predict with high accuracy. Also, that precise star which a user rates a restaurant often gives redundant information for the sentiment analysis. To reduce the redundant information and train better models, user preference is used in substitute of the five-star rating. “92% of users will use a local business if it has at least a 4-star rating”. (Saleh, 2015) Thus, the stars of four and five often reveal the user preference for the restaurant, while stars of one to three means dissatisfaction. It gives an imbalanced data and the ratio of class 1 to class 0 instances is around 3:1. With such conversion, the models can detect a user’s positive or negative attitude toward the restaurant solely by the text content of the review submitted to Yelp. A binary sentiment analysis problem is defined.

4.1.2 Text review conversion

The collection of text reviews cannot be directly used for the machine learning algorithms. Fixed-length vectors of numbers are required as the input for the algorithms. Text feature extraction can tokenize the collection of text documents, learn the vocabulary, and encode new documents.

Bag-of-Words (BoW) model is a common method for text feature extraction. It represents the text in the form of word occurrences within a document, by designing a vocabulary of known words and measuring the presence of those known words. The model discards the order and structure of words in the document and is only concerned with the word occurrence. (Brownlee, 2017)

The common length of the text reviews can help determine how many features (number of words) to be extracted rather than looking at every single word appears in the documents. Both training and testing dataset are used to look at the frequency distribution of text review length. Most of the total reviews have the text length within 2000 words (93rd percentile). (See Table 4.1, Figure 4.1) And it is true across the different star ratings. (See Figure 4.2) Therefore, it is reasonable to limit the number of features to learn to be 2000.

Count	32637
Mean	861.93
STD	702.88
Min	21
25%	371
50%	675
75%	1130
93.3%	2000
Max	5106

Table 4.1 Summary Statistics for Text Length

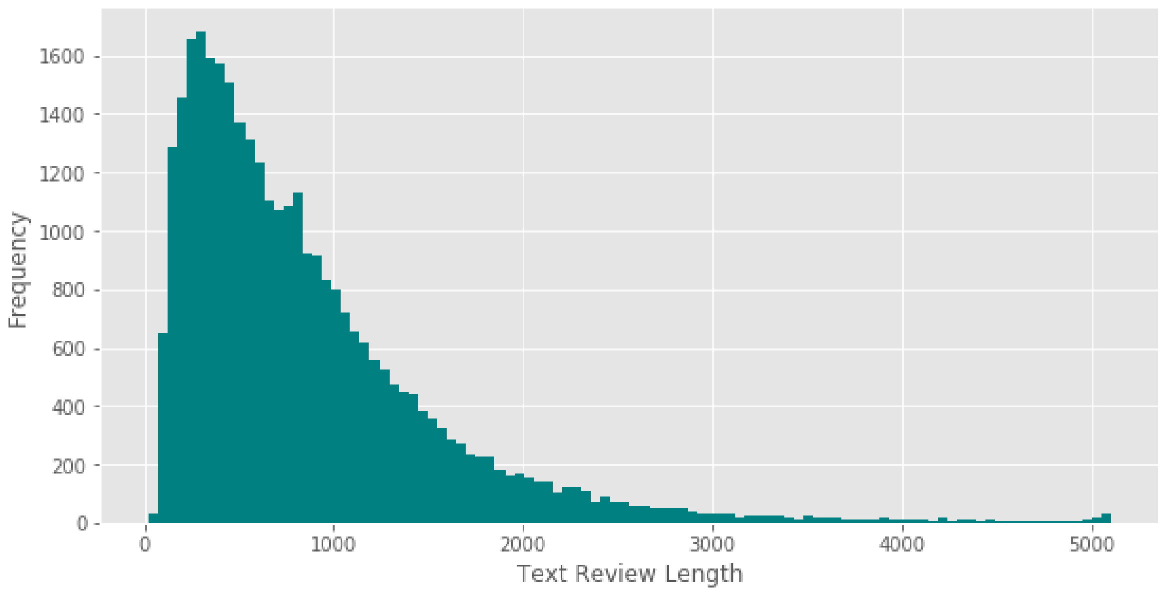


Figure 4.1 Frequency Distribution of Text Review Length

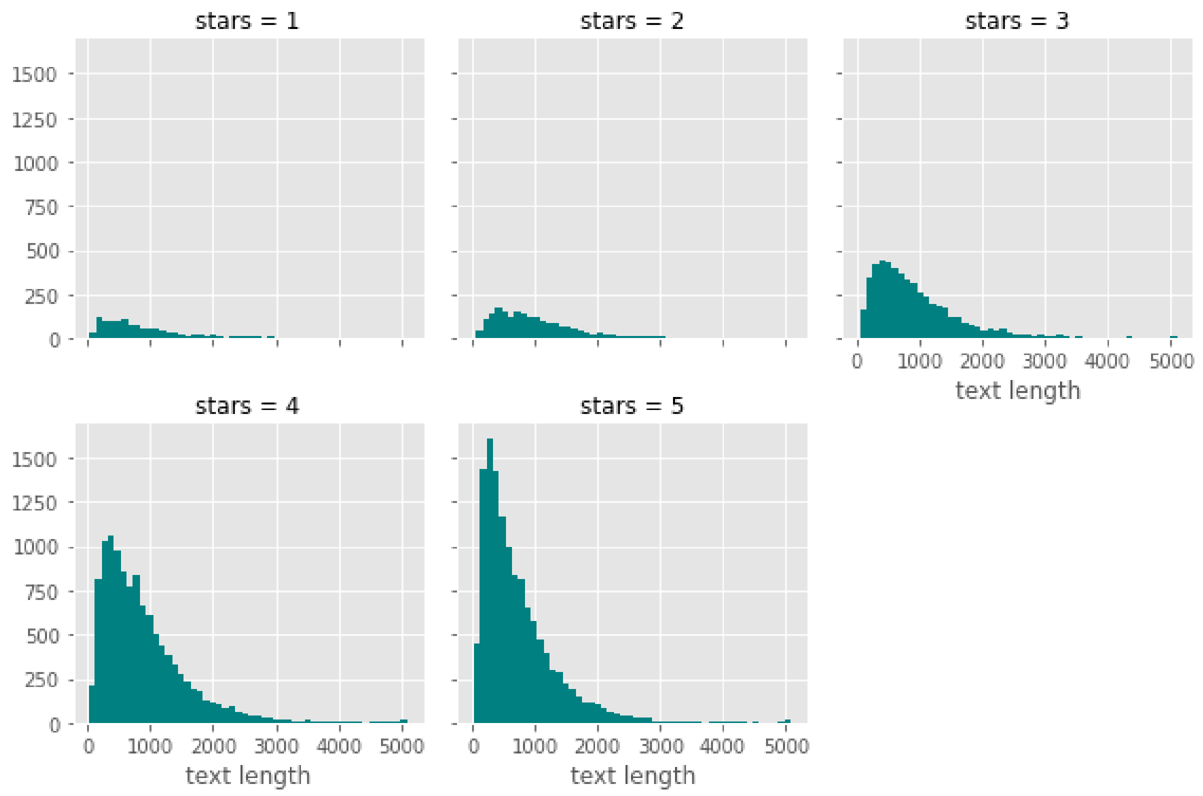


Figure 4.2 Frequency Distribution of Text Review Length based on the star ratings

The scikit-learn library provides several schemes for Bag-of-Words models. The model encodes raw documents of review texts to a matrix of features. It learns vocabulary and features from the training dataset and then transform the testing dataset.

CountVectorizer is a basic vectorizer used to measure word presence by simply counting the number of occurrences in the document. Raw count is the simplest choice of term frequency. Denote term frequency as $tf_{t,d}$, with subscripts denoting the term and then document respectively. This implementation produces a sparse matrix of counts.

Word occurrence is simple but when words appear frequently in too many documents, the large number of word occurrence may not mean that much in the encoded vectors. TF-IDF (Term frequency-inverse document frequency) is another way to evaluate word importance in a document. The resulting score of each TF-IDF feature in the encoded matrix is the product of both term frequency and inverse document frequency. The inverse document frequency is “the logarithmically scaled inverse fraction of the documents that contain the word”. (Srinivasa-Desikan, 2018) It downscales the terms with high occurrence across documents. Document frequency df_t is the number of documents in the collection that contain a term t. Denoting the total number of documents in the collection as N, define the inverse document frequency as follows:

$$idf_t = \log \frac{N}{df_t}$$

Thus, the TF-IDF score of term t in document d is defined as:

$$tf\text{-}idf_{t,d} = tf_{t,d} \times idf_t$$

The proportional weight to term counts and the adjustment for terms that appears frequently, in general, makes TF-IDF the most popular term-weighting scheme. (Manning, Raghavan & Schütze, 2008)

This analysis mainly uses TF-IDF vectorizer. But raw word count vectorizer is used in some cases to compare with TF-IDF vectorizer.

4.2 Classification and Evaluation

In Section 4.1.2, uneven classes (around ¾ data are a positive class) are defined. For each algorithm, use both accuracy and F1 score as the evaluation metrics to have good interpretability and adjustment for the imbalanced dataset. ROC curve and AUC score are not used since the relatively small negative class (dissatisfaction) is not a big concern in the analysis. Cross-validation (cv = 5) is used for algorithm with low computation costs.

For each table of scores in this section, higher scores are highlighted.

The logistic regression classifier with TfidfVectorizer, either cross-validated or not, outperforms among all classifiers. The next best is the linear support vector classifier with TfidfVectorizer.

4.2.1 Logistic Regression

	Accuracy	F1 Score
CountVectorizer	0.8655	0.9110
TfidfVectorizer	0.8709	0.9164

Table 4.2 Logistic Regression with Train/ Test Split

	Accuracy	F1 Score
CountVectorizer	0.8595	0.9068
TfidfVectorizer	0.8730	0.9176

Table 4.3 Logistic Regression with Cross-Validation

4.2.2 Naïve -Bayes

Use cross-validated grid search to find the best parameter combinations based on accuracy and F1 score. The resulting classifier has *alpha = 0.00001*.

	Accuracy	F1 Score
CountVectorizer	0.8430	0.8935
TfidfVectorizer	0.8239	0.8924

Table 4.4 Naïve-Bayes with Train/ Test Split

	Accuracy	F1 Score
CountVectorizer	0.8474	0.8961
TfidfVectorizer	0.8246	0.8925

Table 4.5 Naïve-Bayes with Cross-Validation

4.2.3 Support Vector Machine

	Accuracy	F1 Score
LinearSVC	0.8680	0.9130
Polynomial SVM (gamma = 2)	0.8110	0.8856

Table 4.6 Support Vector Machine with TfidfVectorizer

	Accuracy	F1 Score
CountVectorizer	0.86315	0.9095
TfidfVectorizer	0.8680	0.9130

Table 4.7 Linear Support Vector Classifier

4.2.4 Random Forest

parameters : (n_estimators=100,max_depth=40,min_samples_leaf=10)

	Accuracy	F1 Score
CountVectorizer	0.8061	0.8836
TfidfVectorizer	0.81577	0.8886

Table 4.8 Random Forest

4.3 Other NLP Concerns

This section introduces a few NLP possibilities to experiment text reviews in vector space and get useful information in normalized space. The comparison of vectorizers is based on evaluating previous binary classifiers. The top words and similar reviews in terms of sentiments complement the human-level language processing of text reviews by machines.

4.3.1 CountVectorizer vs. TfidfVectorizer

In Section 4.1.2 two vectorizers with different weighting schemas in scikit-learn library are introduced. It seems that TDIDF is superior with adjusting the number of word occurrences by how few the word occurs across the collection. But is TDIDF vectorizer always better than raw word count vectorizer?

The answer is proved to be “No”. While all other algorithms perform better with TfidfVectorizer, the Navie-Bayes classifier in this analysis show opposite preference in vectorizer selection. Navie-Bayes classifier has both higher accuracy and F1 score with

CountVectorizer for both 70/30 train/ test split and cross-validation. (See Table 4.4 and Table 4.5)

Based on Table 4.9, the above relative performance of vectorizers based off Naïve-Bayes classifier remains consistent for cross-validation across different folds. Thus, the above preference is reliable. TfidfVectorizer should not be considered as a superior choice over CountVectorizer in all cases.

	Accuracy	F1 Score
CountVectorizer	0.8449, 0.8529, 0.8444, 0.8398, 0.8549	0.8942, 0.8996, 0.8942, 0.8911, 0.9012
TfidfVectorizer	0.8247, 0.8245, 0.8273, 0.8190, 0.8273	0.8924, 0.8924, 0.8941, 0.8893, 0.8942

Table 4.9 Naïve-Bayes - Cross Validation Scores Across Folds

4.3.2 Top Words for Classification

The sentiment analysis is to predict the user preference by the text contents of reviews. Besides the model accuracy scores, it may be interesting to look at a collection of the specific words that in general (not specific to a user or a business) are more often related to a positive or a negative predicted rating.

The choices of vectorizer differ in defining the score assigned to each word in the encoded vectors and will produce different collections. Thus, both CountVectorizer and TfidfVectorizer are used to compare the results.

Logistic regression classifier is an ideal model to train for this purpose due to good interpretability of coefficients. The coefficients of each word (feature) in the fitted model implies not only a positive or negative relationship but also relative importance.

The top 20 words that lead to positive or negative rating predictions with the logistic regression classifier are shown in Table 4.1.

	Positive Prediction	Negative Prediction
CountVectorizer	'gem', 'disappoint', 'heaven', 'phenomenal', 'pleasantly', 'olives', 'kinds', 'perfection', 'nut', 'efficient', 'affordable', 'caramel', 'generous', 'suggestion', 'perfectly', 'thankfully', 'die', 'perfect', 'creative', 'awesome'	'mediocre', 'horrible', 'worst', 'meh', 'soggy', 'disappointing', 'uncomfortable', 'gross', 'overpriced', 'sick', 'lacked', 'worse', 'bland', 'awful', 'sad', 'memorable', 'tough', 'ok', 'alright', 'hoping'
TfidfVectorizer	'delicious', 'amazing', 'perfect', 'great', 'awesome', 'love', 'favorite', 'definitely', 'excellent', 'loved', 'perfectly', 'best', 'perfection', 'disappoint', 'fantastic', 'yummy', 'friendly', 'recommend', 'bomb', 'glad'	'ok', 'mediocre', 'bland', 'okay', 'horrible', 'dry', 'wasn', 'disappointing', 'worst', 'average', 'meh', 'unfortunately', 'overpriced', 'disappointed', 'terrible', 'bad', 'maybe', 'sad', 'awful', 'soggy'

Table 4.10 Top 20 Words for Positive or Negative Predicted Ratings

4.3.3 Similar Review Search

With the sparse encoded text review data, it is easy to find similar reviews by comparing cosine similarity. Cosine similarity calculates the cosine of the angle between two documents one the vector space. It measures the orientation rather than magnitude. Thus, the similarity on vector space can be used to compare between documents on normalized spaced. (Perone, 2013)

For a random review in the test data, find the top few reviews in training dataset with the highest cosine similarity. The associated ratings are easy and efficient to get and implies the content similarity that can take tremendous time using the human-reading method. Table 4.11 draws a sample review and the top 5 similar reviews of it are searched and shown in Table 4.12.

The similar review search engine can somehow predict what star rating is given to the review by looking at star ratings for most similar reviews. Predicting positive or negative sentiments with the simple average of top five similar reviews gives an accuracy score of 70%. Around 20% of reviews are falsely rated as negative, and 10% falsely rated positive. While the prediction performance falls far behind with previous binary classifiers, it gives some supports for the similar review search engine.

Star Rating	Text Review
5	"Food was fantastic and we had the most attentive server. We had an omakase and man was it awesome. They had a badass beer selection. The omokase had everything. Their best cold plates and hot plates and some of the best otoro I've had in the City. The Kobe beef was also absolutely delectable. The food quality was as top notch as the elite and did not break the bank."

Table 4.11 A Sample Review in Testing Dataset

Top	Star Rating	Text Review
1	5	"I've been here multiple times now for both dinner and just a few drinks at the bar. The food is totally fantastic and you get so much (which is good because the prices are on the high side). The beer selection is phenomenal as both my husband and I love beer. Their happy hour isn't the best but overall selection of food is great. Additionally if you're vegan they have plenty of menu time options!"
2	5	"Absolutely fantastic! \r\r\n\r\r\nEverything about this place was spot on. The parking was easy and plentiful. The decor is great. The service was fantastic. The food was phenomenal! \r\r\n\r\r\nThe food is not only good but it's plentiful. You will either be rolling out of the restaurant or you will have leftovers. \r\r\n\r\r\nI would absolutely order the lamb chops. These were probably the best I've ever had. \r\r\n\r\r\nDefinitely come here for some good food!"
3	4	'Came here for a cold beer and a small bite to eat. The staff was friendly, beer ice cold and food was good. I would come here again.'
4	5	'I long time Las Vegas Mexican Restaurant, always has good food, cold beer and the best servers!\r\r\n\r\r\nI really enjoy the beef tacos'
5	5	"Awesome idea! My mother and I had a fantastic time at Chubby Cattle! The service was phenomenal and the experience was just so much fun and like nothing I've ever done before!\r\r\n\r\r\nYou can eat as much or as little as you like. Just choose your hot pot broth, select some meats of the menu, grab some sauces from their sauce bar and pick and choose the plates as they roll by on the conveyor belt. Hungry for more? Then just keep grabbing those plates and they'll tally the plates up at the end. (If you're wondering how much each plate costs, just check out their little plate key below every window). \r\r\n\r\r\nMy mother is in from Texas and she loved the experience and can't wait to go back. Besides just having a great time, the staff were all very helpful and super nice and attentive. Kind of rare for a Chinatown restaurant to be honest so it was greatly appreciated. \r\r\n\r\r\nWould I go back? Absolutely without a doubt! Would I recommend it to my friends and family? Heck ya!"

Table 4.12 Top 5 Similar Reviews for Sample Reviews in Training Dataset

CHAPTER 5

Recommendation Systems

The sentiment analysis in Chapter 4 focus only on relationships between review contents and corresponding rating/ preference. The two involved entities, reviewers (users) and businesses (items) are ignored. As the encoded vectors of review documents are the sole input for the algorithms, a preference prediction requires that the user provides a review for the business. An explicit star rating is always given with a Yelp review. But how to predict the positive or negative attitude of a user towards a business that the user has not reviewed yet?

The new question leads to the popular discussion of recommendation systems. For a user, algorithms predict the attitudes for businesses and find the unreviewed businesses with the highest predicted rating as the recommendation.

Neighborhood-based collaborative filtering (also known as a memory-based algorithm) is based upon the fact that similar users and items have similar patterns in rating. User-based and item-based collaborative filtering are two primary types of neighborhood algorithms. The differences lie in the neighborhood choosing and rating predicting. The user-based approach defines the neighborhood by similarity among users, and the rating is predicted by ratings of neighboring users for the same business. The item-based approach defines the neighborhood by similarity among items (businesses), and the rating is predicted by ratings of the user for neighboring businesses. (Aggarwal, 2016) In this chapter, an item-based collaborative filtering recommender is built steps by steps from a baseline system.

5.1 Data Implementation

The simple binary conversion of the five-star rating for sentiment analysis (Section 4.1.1) is good since the classifier only focuses on predicting preference from text content, regardless which user or business is involved. However, in order to get a personalized recommendation, user preference specific to that user rather than in general should be predicted. Different users may have different rating standards. Some users may be very critical and reluctant to give a perfect “5” star. Some may be very sweet and tolerant with a little imperfection in rating restaurants. Thus, normalization adjusted by each user’s rating performance is needed.

Define a user-specific preference score by subtracting each actual star rating by the average rating per user. It implies the biases of users. Figure 5.1 Shows the distribution of the new target variable. The conversion changes discrete values into decimal values in a continuous space. The user-specific preference is more comparable across reviews considering difference users’ rating standards.

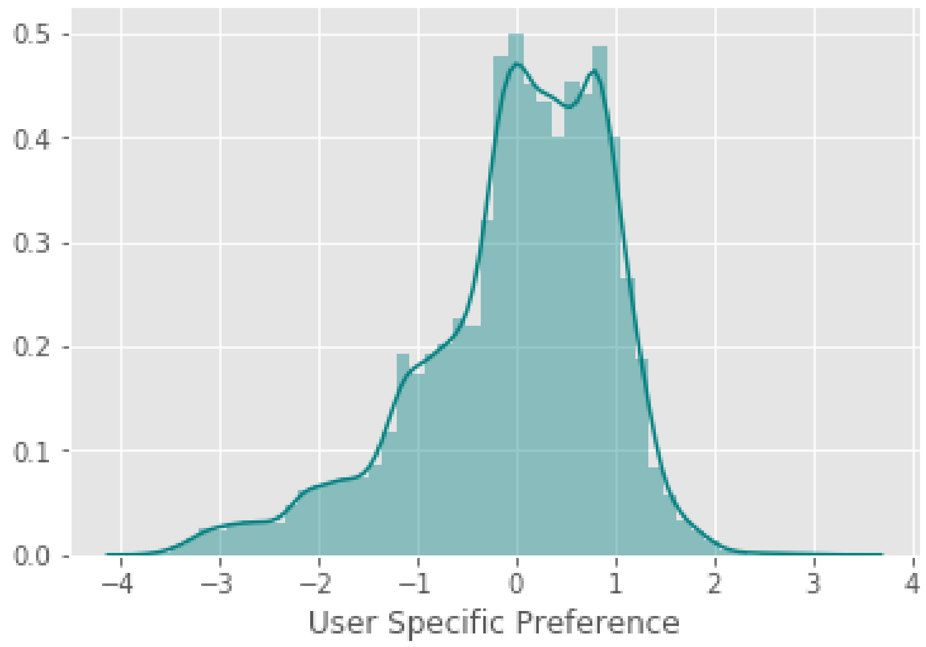


Figure 5.1 Distribution Plot of User Specific Preference

5.2 Recommender Buildling

A simple and intuitive baseline recommender is to consider both user and item (business) biases. For the rating of user u given to business b has a baseline estimate defined as below.

$$\hat{Y}_{ub}^{baseline} = \bar{Y} + (\bar{Y}_u - \bar{Y}) + (\bar{Y}_b - \bar{Y})$$

$\bar{Y} = average\ rating\ across\ all\ reviews$

$\bar{Y}_u = average\ rating\ of\ the\ user\ u$

$\bar{Y}_b = average\ rating\ of\ the\ business\ b$

The estimated rating is composed of global average and divergence cause by user-specific and item-specific bias in rating. It focuses on the user u, business u and average performance only.

To better predict the rating, the estimate can incorporate ratings of similar users for similar businesses.

$$\hat{Y}_{ub} = \hat{Y}_{ub}^{baseline} + \frac{\sum_{i \in L^k(b)} S_{bj}(Y_{ui} - \hat{Y}_{ui}^{baseline})}{\sum_{i \in L^k(b)} S_{bi}}$$

$L^k(b) = k\ neighborhood\ items\ of\ business\ (item)\ b$

$S_{bi} = similarity\ measure\ between\ business\ b\ and\ business\ i\ in\ L^k(b)$

To get S_{bi}, a database of similarity for each pair of businesses based on common reviewers' rating is needed. The number of common reviewers of two businesses is the support. Pearson correlation coefficient of the user specific preference given by common reviewers gives the similarity score. Only common reviews matter in this step. Note the support is quite low across all business pairs. (See Figure 5.2) The mean support is 2.66.

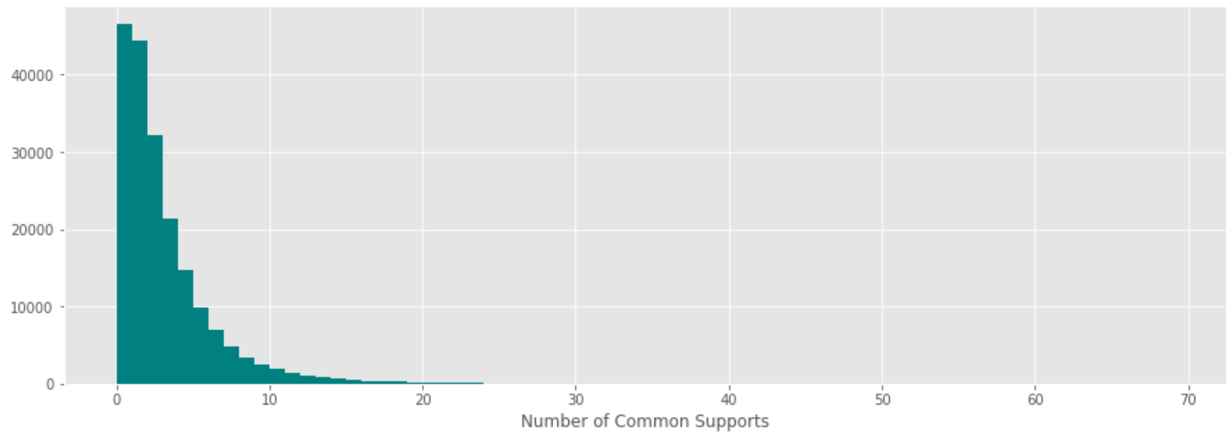


Figure 5.2 Number of Common Supports

To get the k nearest businesses for one business b, the similarity needs to be adjusted by the number of common supports before ranking. A regularization parameter ($reg = 3$) is used.

$$S_{mi} = \frac{N_{common}\rho_{mi}}{N_{common} + reg}$$

$N_{common} = number\ of\ common\ supports$

$\rho_{mi} = Pearson\ similarity\ of\ business\ m\ and\ i$

With the ability to find a list of most similar businesses ($L^k(b)$)for a business b, an item-to-item recommendation system can be built. For a user u, find c businesses that are highest rated by the user. And for each of those businesses, find their k nearest neighbors. Duplicates and the business that user u has reviewed needs to be removed from the list of $c \times k$ neighbor businesses. From the reduced list, recommend businesses with the highest business average ratings.

The above recommendation system compares the neighbor businesses of user u's favorite choices solely by the item itself ($\bar{Y}_b$) without considering the user. An improved model can thus be proposed. When comparing neighbors of top choices, use imputed ratings adjusted by how user u rates the neighbors. $L^k(b;u)$ is used instead of $L^k(b)$ in the formula.

$$\hat{Y}_{ub} = \hat{Y}_{ub}^{baseline} + \frac{\sum_{i \in L^k(b;u)} S_{bj}(Y_{ui} - \hat{Y}_{ui}^{baseline})}{\sum_{i \in L^k(b;u)} S_{bi}}$$

$L^k(b;u) = k\ neighbor\ items\ of\ business\ b\ that\ has\ been\ rated\ by\ user\ u$

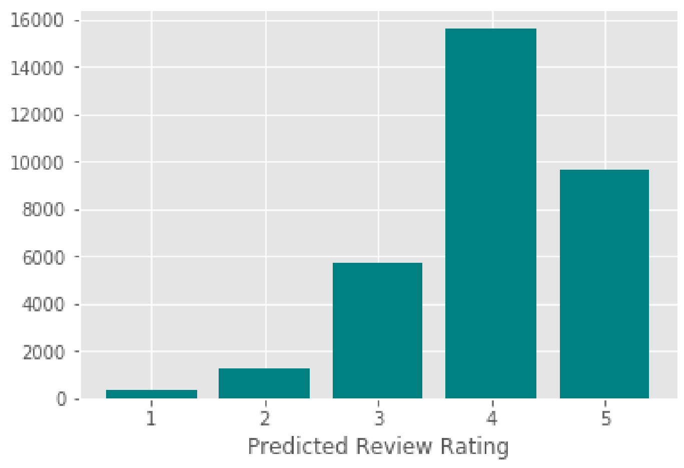
A final step is to convert the predicted rating $\hat{Y}_{ub}$ back to a five-star rating. $\hat{Y}_{ub}$ is rounded to the nearest integer. And if the value is greater than 5 or less than 1, convert the value to 5 or 1 respectively. This makes the predicted rating generated more interpretable and comparable to actual rating since a user can only rate with the five-star scale.

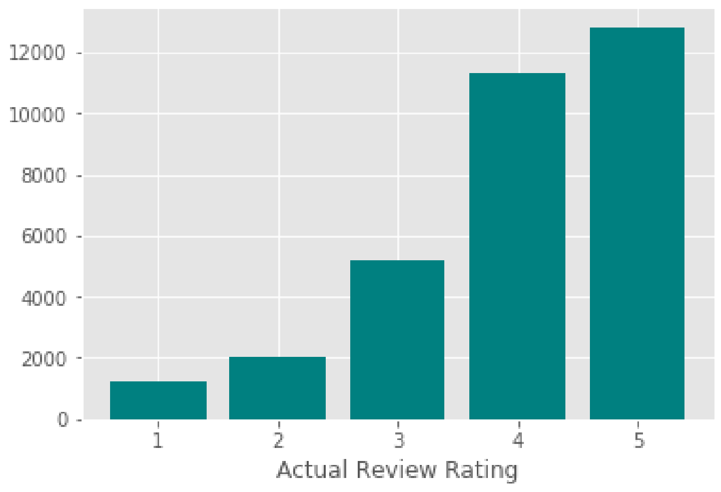
5.3 Error Analysis

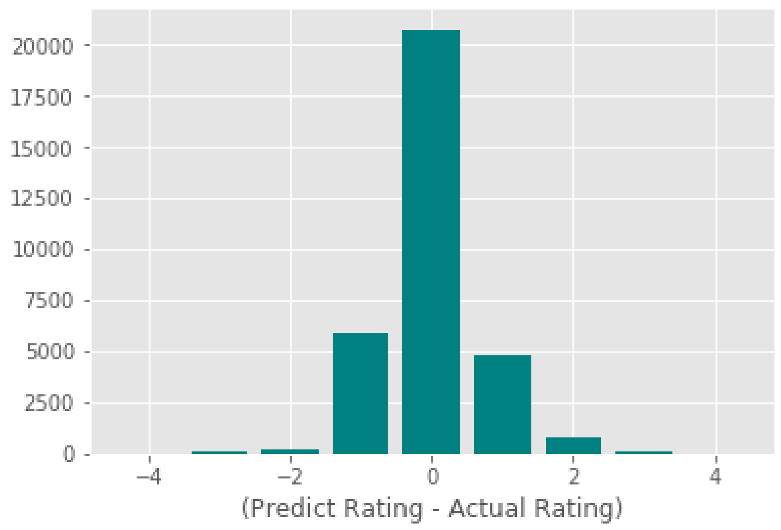
As the whole dataset gives quite low common supports for business pairs, it implies the dataset can be relatively small for the recommendation system. Thus, no test/ train split or cross-validation is used. Error analysis comes from testing on the training dataset alone to provide some insights.

With the established formula, a predicted five-star rating in each review of business b by user u can be imputed. Compared to actual ratings, the predicted ratings converge to the average ratings across all users, businesses, and reviews. See Figure 5.3 and Figure 5.4 for a detailed comparison.

The established recommendation system has an RMSE of 0.71174. Among all reviews, 17.52% of them have a higher predicted rating than the actual rating, 18.85% have lower predicts. The accuracy score of the recommender predicting five-star rating is 0.63627, which is decent.







mean	0.008978
std	0.711693
min	-4
max	4

Figure 5.3 Bar Plots of Ratings

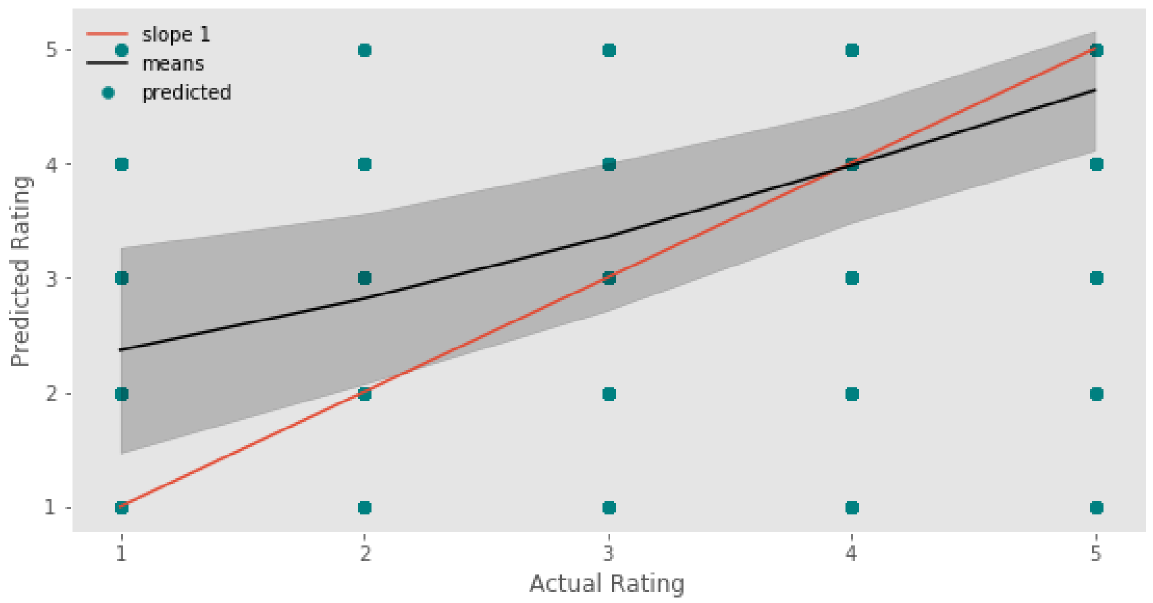


Figure 5.4 Predicted Rating vs. Actual Rating

CHAPTER 6

Conclusion and Further Interests

This paper explores two ways in which online reviews influence the consumers' purchase behaviors. Consumers search the business of interest, read posted reviews and perceive sentiments to form their judgements on business performance. The review websites predict how users would rate different businesses and recommend to consumers the businesses they might rate high. Thus, sentiment analysis on a per review basis and recommender building based on user-business pairs are main approaches to rating predictions in this paper.

Natural Language Processing stimulates the human-level understandings of the text reviews. In sentiment analysis, important text features are extracted with vectorizers and the encoded documents are used to train binary classifiers to get positive (rating star of 4 or 5) or negative (rating star below 4) rating predictions. TfidfVectorizer is a better vectorizer in most cases. As for classification algorithms, logistic regression classifier (Acc =0.873, 0.918) and linear support classifier (Acc = 0.868, F1 =0.913) have the best trained models. The encoded documents can also be used to find top words for positive or negative predictions and search for similar reviews across documents.

The neighborhood collaborative filtering recommender built in this paper gives user-specific rating prediction. It predicts five-star rating by nearest item neighbors and common reviewers' ratings. Ranking predictions can give top recommendations for a user. Testing on the whole dataset, the built recommender gives decent accuracy measures (Accuracy = 0.63627, RMSE = 0.71174).

Although sentiment analysis and the recommender system explores reviews from different perspectives, the comparisons can still be made to provide some hints. Binary sentiment analysis is on a per review basis and gives seemly higher accuracy measures. And it also adds on real-world interpretability of review ratings by learning from word features. However, the models can only tell positive or negative sentiments in reviews instead of five-star rating estimates. The neighborhood-based recommender is modeled to get the five-star predictions specific to a user and a business. The predictions can be directly compared to actual ratings with decent accuracy measures. And the user-specific recommendations can be made.

For further studies based on this paper, there are three possible ways to improve on current models. First, a hybrid prediction model can be built by incorporating both text and user-item features which are discussed separately in this paper. Second, more user-item features can be included in the model. An example can be considering the preferences of users' social relations (e.g. Yelp friends) as more relevant. (Qian, Feng, Zhao & Mei, 2014) Lastly, other advanced algorithms can be introduced. For example, the current collaborative filtering recommender can consider matrix factorization or deep learning approaches such as neural collaborative filtering (He et al., 2017) for comparison.

REFERENCES

Afonja, T. (2017). Accuracy Paradox. Retrieved from https://towardsdatascience.com/accuracy-paradox-897a69e2dd9b

Agarwal, A., & Chauhan, M. (2017). Similarity Measures used in Recommender Systems: A Study. *Nternational Journal Of Engineering Technology Science And Research*, *4*(6).

Aggarwal, C. (2016). *Recommender Systems: The Textbook*. Switzerland: Springer International Publishing.

Brownlee, J. (2017). A Gentle Introduction to the Bag-of-Words Model. Retrieved from https://machinelearningmastery.com/gentle-introduction-bag-words-model/

Brownlee, J. (2017). How to Prepare Text Data for Machine Learning with scikit-learn. Retrieved from https://machinelearningmastery.com/prepare-text-data-machine-learning-scikit-learn/

Dharaneeshwaran, N., Nithya, S., Srinivasan, A., & Senthilkumar, M. (2017). Calculating the user-item similarity using Pearson's and cosine correlation. *2017 International Conference On Trends In Electronics And Informatics (ICEI)*. doi: 10.1109/icoei.2017.8300858

Garg, R. (2018). 7 Types of Classification Algorithms. Retrieved from https://www.analyticsindiamag.com/7-types-classification-algorithms/

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. (2017). Neural Collaborative Filtering. *Proceedings Of The 26Th International Conference On World Wide Web - WWW '17*. doi: 10.1145/3038912.3052569

Le, J. (2018). The 4 Recommendation Engines That Can Predict Your Movie Tastes. Retrieved from https://medium.com/@james_aka_yale/the-4-recommendation-engines-that-can-predict-your-movie-tastes-bbec857b8223

Manning, C., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. New York: Cambridge University Press.

Nandi, M. (2017). Recommender Systems through Collaborative Filtering. Retrieved from https://blog.dominodatalab.com/recommender-systems-collaborative-filtering/

Perone, C. (2013). Machine Learning :: Cosine Similarity for Vector Space Models (Part III) | Terra Incognita. Retrieved from http://blog.christianperone.com/2013/09/machine-learning-cosine-similarity-for-vector-space-models-part-iii/

Qian, X., Feng, H., Zhao, G., & Mei, T. (2014). Personalized Recommendation Combining User Interest and Social Circle. *IEEE Transactions On Knowledge And Data Engineering*, *26*(7), 1763-1777. doi: 10.1109/tkde.2013.168

Ridwan, M. (2014). Predicting Likes: Inside A Simple Recommendation Engine's Algorithms. Retrieved from https://www.toptal.com/algorithms/predicting-likes-inside-a-simple-recommendation-engine

Saleh, K. (2015). The Importance Of Online Customer Reviews [Infographic]. Retrieved from https://www.invespcro.com/blog/the-importance-of-online-customer-reviews-infographic/

Srinivasa-Desikan, B. (2018). *Natural language processing and computational linguistics*. Packt Publishing.

Swalin, A. (2018). Choosing the Right Metric for Evaluating Machine Learning Models — Part 2. Retrieved from https://medium.com/usf-msds/choosing-the-right-metric-for-evaluating-machine-learning-models-part-2-86d5649a5428

Tripathy, A., Agrawal, A., & Rath, S. (2015). Classification of Sentimental Reviews Using Machine Learning Techniques. *Procedia Computer Science*, *57*, 821-829. doi: 10.1016/j.procs.2015.07.523

Wang, B. (2018). Python Implementation of Baseline Item-Based Collaborative Filtering. Retrieved from https://medium.com/@wwwbbb8510/python-implementation-of-baseline-item-based-collaborative-filtering-2ba7c8960590

Yelp Dataset. (2018). Retrieved from https://www.yelp.com/dataset/challenge