Rating Prediction of Google Local Reviews

Shiyi Hua
Business Analytics
UC San Diego
s8hua@ucsd.edu

Akshay Kotha
Business Analytics
UC San Diego
akotha@ucsd.edu

Ziyuan Yan
Business Analytics
UC San Diego
ziy062@ucsd.edu

ABSTRACT

When people utilize online evaluations to make decisions, they need to look at various factors such as star ratings, review texts, locations, etc. Often times it is difficult for users to read through all reviews, decide which factors are important to them, and incorporate the related information to make final decisions. User experience would be highly improved if there is a recommender system that predicts ratings given a user and a place. In this paper, we report on our approaches to build such a recommender system. We proposed regression-based recommendation measures, which take into account the textual component of user reviews as well as other related factors. We also tried to propose a KNN algorithm that would generate personalized recommendations based on user-item relationships as well as relationships within user and items themselves.

1 INTRODUCTION

Most people like to use online reviews as a reference when they make decisions about dining at a restaurant, purchasing an item, watching a movie, etc. With the boom of the internet and social media, people are more and more willing to share their opinions and reviews online, which in turn creates a huge amount of information and data for them to choose from. Despite the growing popularity of customers using online information, most websites don't have a suitable recommender system that provides customers with the information they want. Customers still need to browse through each comment and do their own research. User experience would be greatly improved if all the above problems were taken into account.

Incorporating important factors into prediction and modeling the interactions between users and items are the main tasks of a recommender system. The goal of this project is to explore the Google Local dataset and develop a decent model for a recommender system. Our work takes the approach of combining regression-based measures, machine learning, collaborative filtering, personalized recommendations, and text analysis to develop an ideal recommender system.

The report is structured as follows. The data part includes an overview of the dataset, and how we clean, explore and transform it. Then we introduced some works that are related to this topic. The prediction part includes the text analysis we did, different models we approached, and our thoughts about these methodologies. The conclusion summarizes our work and suggests possible improvements.

2 STRUCTURE IDENTIFICATION AND ANALYSIS

2.1 Datasets Overview

We used the Google Local Reviews dataset, which contains User Data (178mb), Places Data (276mb), and Review Data (1.4gb). The dataset includes detailed place name, price, address, hours, phone, closed, gPlusPlaceId, gps, every user's rating, reviewText, categories, unixReviewTime, etc. It has a total of 4,567,431 users, 3,116,785 local businesses, 11,453,845 reviews and ratings, as well as 48,013 different categories of business. Our goal is to predict ratings using the different features in this dataset.

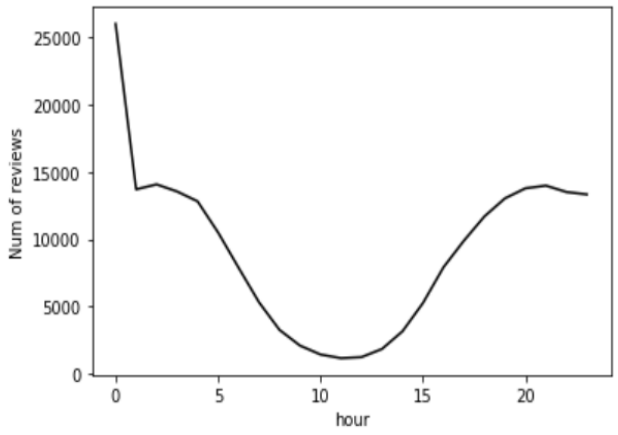
2.2 Data Cleaning

The original dataset contains a huge amount of data all over the world, which causes two problems for our further analysis. The first is that due to limited computational power, it's difficult for us to deal with all the data. The other problem is that we want to limit the language to English so that our text analysis will be more precise. We also want to combine information into one dataset, so that we don't need to switch between the datasets when doing analysis. Therefore, the following procedures are performed to clean our dataset:

1. Filter all Places in California with non-empty gps and price (GPS coordinates: Latitude: 32°32′ N to 42° N, Longitude: -124°26′ W to -114°8′ W). Now we have 48665 places.
2. Use the first 4 million reviews data, keep only the reviews that have the gPlusPlaceId in California Place dataset.
3. Assign gps and price to each review so that we have all the information that will be used in one dataset.
4. Filter out everything that is None or empty.

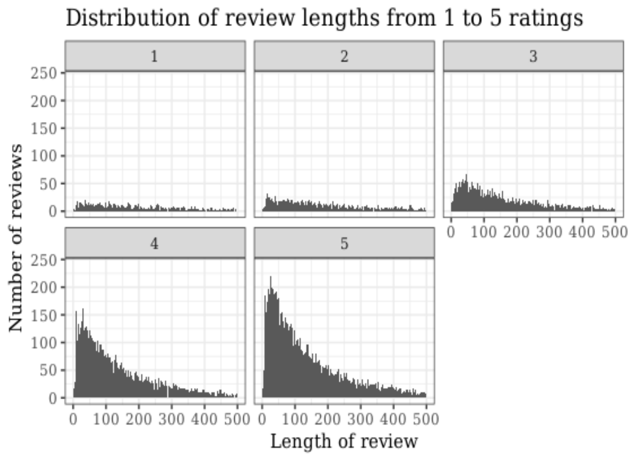
After data cleaning, we have a total of 76254 reviews and 48865 places.

2.3 Data Visualization



Number of Reviews vs. Hour

Observations: Number of reviews is highest at around midnight (12am) and lowest at around noon (12pm). Local maximum is around 2am and 8pm. There is a strong relationship between hour and number of reviews.

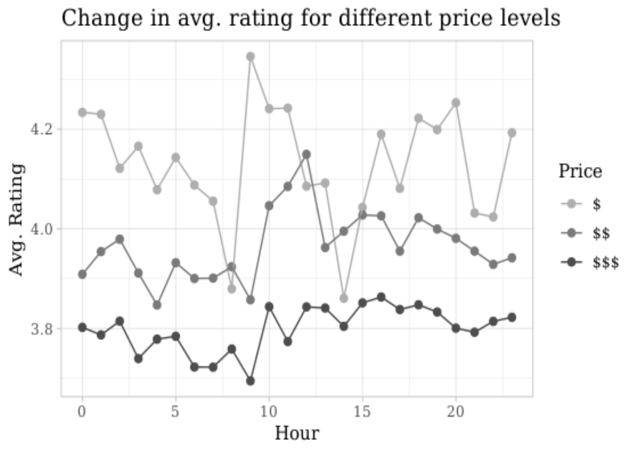


Number of Reviews vs. Length of Reviews, facet by Star Rating

Observations:

1. Very few people give low star ratings (ratings = 1 or 2)
2. People tend to write reviews with a review length of 50~100 words.
3. For reviews that have a rating of 1 or 2, review length varies more than those that have ratings of 3~5.

There is a strong relationship between star rating and review length.



Average Rating vs. Hour, facet by Price

Observations:

1. Average ratings tend to be higher for stores that have a lower price level. On the contrary, average ratings tend to be lower for stores that have a higher price.

There is a strong negative relationship between price and rating.

2. Average ratings are lowest around 8-9am and 1-2pm.

There is a strong relationship between hour and rating.

2.4 Data Transformation

After data exploration, we want to add hour, review length and price level to the dataset for the prediction task. Therefore, the following procedures are performed to transform our cleaned dataset:

1. Using UnixReviewTime, retrieve time in the format of "year-month-day hour:minute:second". Save hour to the dictionary as a feature.
2. Add review length to the dictionary as a feature.
3. For each price level ($, $$ or $$$), save to the dictionary using ordinal-encoding as a feature.

3 RELATED WORK - LITERATURE

As our goal was to learn something new about rating prediction, unlike rating prediction with latent factor models, we implemented review-based rating prediction and it's variants. We started with average prediction for that place and used sentiment based prediction as quoted in [1] and worked on improving our model with sentiment based rating. The idea of personalized recommendations where using KNN was also agreed upon by our group. As we studied in depth, we also found some interesting implementations where rating was not used instead a new text-based score was calculated based on features of the text. Then this was scaled down to the scale of actual rating and predictions were made. It's interesting that as we delve deeper into extracting information from text, the models had better performance with lower MSE. Obviously, the models with unscaled text rating had a bit higher MSE which still talks for an optimal performance when scaled it to the order of actual ratings.

4 REVIEW TEXT ANALYSIS

We would like a fixed-dimensional representation of documents. In our case, we would like to describe reviews using feature vectors. This would allow us to compare reviews written by users about different places, and associate weights with particular features to solve predictive tasks [5]. For 76,254 reviews, we removed capitalization and punctuation for each review text. We also excluded stop words like "me", "is", and "with". As a result, we have 1,623,179 words in total and 60,413 distinct words along with the number of appearance of each word. Then we sorted these distinct words from the most number of appearance to the least, and chose the first 1,000 as the popular words, which are randomly stored in a set. We assigned index to each popular word and zipped them in a dictionary. It turns out that the most popular word is "food" that appears 32,377 times, and the least one among the first 1000 popular words is "forward" that appears 263 times. This analysis of words in each review text helps us to build a feature vector for rating prediction.

5 RATING PREDICTION

Our goal is to build a predictor to estimate the rating that each user gives to a place on Google Local based on statistically significant and relevant features. We applied three main methodologies to see the performance of each model.

5.1 Linear Model With Ridge Regression

5.1.1 Baseline Model

We first trained a regressor based on the Review Text Analysis.

$$Rating = \theta_0 + \theta_1 * CountPopularWords$$

We created a feature vector of length 1,001 for each review. 1 is the intercept (i.e. global mean) and the remaining 1,000 is set as 0's first corresponding to the index of sorted 1,000 popular words. For each review, we broke down the review text into words. We added 1 to the corresponding indexes every time when the words appear in the popular word set. Thus, our feature vector is a 76,254 * 1,001 matrix, and our label is a 76,254 * 1 rating matrix.

Then we divided the feature and label into training, validation, and testing sets with the ratio of 80%:10%:10%. We trained the linear model with ridge regression on the training set, tuned model parameters on the validation set, and tested the result on the testing set. It turns out that the training MSE (Mean Squared Error) is 0.844967, the validation MSE is 0.888903, and the testing MSE is 0.853999.

5.1.2 Advanced Model 1

We want to improve the baseline model by using a multivariate regression model. Based on the exploratory analysis, we considered that adding features of review length, review hour, and price level might be helpful.

$$Rating = \theta_0 + \theta_1 * CountPopularWords + \theta_2 * ReviewLength + \theta_3 * ReviewHour + \theta_4 * PriceLevel$$

Besides the feature vector of counting popular words, we appended the length of each review; then we one-hot encoded the hour at which each reviewer wrote review down and one-hot encoded the price level of each place as well, and appended them to the feature vector. Specifically, hour varies from 0 to 23, and price level varies from low, moderate, to high, we one-hot encoded these two categorical variables. Right now the feature vector has the length of 1,028. This advanced model 1 yields that the training MSE equals 0.841425, the validation MSE equals 0.885688, and the testing MSE equals 0.851477. All three MSEs are lower than our baseline model.

5.1.3 Advanced Model 2

We desired to further improve the advanced model 1. Since the price is from low to high, we thought whether it is better to ordinally encode price level rather than just one-hot encoding.

$$Rating = \theta_0 + \theta_1 * CountPopularWords + \theta_2 * ReviewLength + \theta_3 * ReviewHour + \theta_4 * OrdinalPriceLevel$$

Compared to the advanced model 1, in the feature vector, we kept the intercept, the count of popular words, the length of review, and the one-hot encoding hour, while we changed the one-hot encoding price level to the ordinal encoding price level. That is, we ordinally encoded low price level as 1, moderate price level as 2, and high price level as 3 instead of one-hot encoding low price level as 1/0/0, moderate price level as 0/1/0, and high price level as 0/0/1. However, this advanced model 2 yields that the training MSE equals 0.841621, the validation MSE equals 0.885732, and the testing MSE equals 0.851857, which are all higher than our advanced model 1.

5.1.4 Advanced Model 3

Although the advanced model 2 is not as good as the advanced model 1, it is better than the baseline model and its results are actually close to the model 2. Thus, we considered keeping the ordinally encoding price level and further dealing with the review hour.

$$Rating = \theta_0 + \theta_1 * CountPopularWords + \theta_2 * ReviewLength + \theta_3 * IntegerHour + \theta_4 * OrdinalPriceLevel$$

We included the intercept, the count of popular words, the length of review, the ordinally encoding price level, and the integer encoding review hour. In specifics, we just changed the string type of review hour to the integer type. Right now the review hour is encoded as integer from 0 to 23. This advanced model 3 yields that the training MSE equals 0.842303, the validation MSE equals 0.884685, and the testing

MSE equals 0.850684. The validation and testing performance are better than the advanced model 1.

5.1.5 Advanced Model 4

Until now the advanced model 3 is the best model, and then we contemplated whether it is possible to modify our feature of counting popular words.

$$Rating = \theta_0 + \theta_1 * CountOnlyPosNegWords + \theta_2 * ReviewLength + \theta_3 * IntegerHour + \theta_4 * OrdinalPriceLevel$$

We excluded neutral words and only kept positive- and negative- sentiment popular words, along with the intercept, the length of review, the integer encoding hour, and the ordinal encoding price level in the feature vector. This advanced model 4 yields that the training MSE equals 0.878442, the validation MSE equals 0.936154, and the testing MSE equals 0.899768. The performances on training, validation, and testing sets are all weaker than the baseline model.

5.2 Sentiment-Based Text Rating

5.2.1 Single Review Text Sentiment-based Model

Intuitively, the sentiment expressed in the textual review would be the best indicator of a user's likes and dislikes. To leverage the sentiment information into a single score for each review, we translated our annotated text reviews into a text rating score that can easily be compared to the metadata star rating [1]. We used TextBlob to directly judge whether each review has a negative, neutral, or positive sentiment. If the polarity of the review by TextBlob is less than 0, then the sentiment is positive; if the the polarity of the review by TextBlob equals 0, then the sentiment is neutral; otherwise, it is positive. Then we one-hot encoded the sentiment of each review.

$$Rating = \theta_0 + \theta_1 * Sentiment + \theta_2 * ReviewLength + \theta_3 * IntegerHour + \theta_4 * OrdinalPriceLevel$$

However, the training MSE equals 1.015468, the validation MSE equals 1.019132, and the testing MSE equals 1.015864. The performances on training, validation, and testing sets are all weaker than the baseline model.

5.2.2 Regression-based Text Rating with Weights

The motivation to this model was the above model in which sentiment of the review text was used as a feature directly. In this model, we used the features [Prop. of positive sentences] and [Prop. of negative sentences] in the review. We tried to make it better by considering assigning appropriate weights to positive and negative parts of the review text. We used a custom function to split review text and TextBlob as a start to find out the polarity of each sentence.

$$Rating = \theta_0 + \theta_1 * PropPosSen + \theta_2 * PropNegSen + \theta_3 * CountPopularWords + \theta_4 * IntegerHour + \theta_5 * OrdinalPriceLevel$$

Note: $\theta_3 * [CountPopularWords]$ doesn't mean we have one θ_3, it in fact means that we have learned coefficients when that popular word is in the review text. The coefficients θ_1 and θ_2 represent the coefficients of positive proportion of sentences and the negative proportion f

All the previously tried models helped us in getting the <u>best</u> training MSE equals 0.80102, the validation MSE equals 0.83651, and the testing MSE equals 0.81599. The performances on training, validation, and testing sets are better than the baseline model and the above model where we fit the sentiment of the review as a whole along with other features (word vector, review length, hour, price).. This slightest improvement could be attributed to assigning weights of positive and negative sentences in the review text and making sure the one-line reviews are detected as positive or negative appropriately.

5.3 Personalized Recommendations - KNN

Here, the idea was basically to find the nearest neighbors (Pearson distance [1]) based on the number of places reviewed by the user and predict the average rating of the nearest neighbors for a {user, place} combination. But in the process of implementing Pearson-based distance calculation, we found out that the initially filtered data for California is very sparse that the top two users with {180, 135} places reviewed have only 6 places in common. Our learning from here was to use a different filter, for instance, filter out the users which have a minimum number of places reviewed so that the data is not as sparse. We wish to implement the same model on this kind of data in the future.

6 EVALUATION OF METHODOLOGY

Section	Models	MSE on Testing Set
5.1.1	Baseline Model	0.853999
5.1.4	Advanced Model 3	0.850684
5.2.1	Single Review Text Sentiment-based Model	1.015864
5.2.2	Regression based text rating with weights to sentence types	0.815991

We formulated the baseline model based on our review text analysis. Then we improved our baseline model by trying to change the way of encoding categorical variables. Furthermore, we switched from counting popular words to encoding the sentiment of one whole piece of review text. It is the linear model with ridge regression that played a primary role in the procedure of optimizing our model for rating prediction. The main reason is that ridge regression can minimize the multicollinearity among features and then lower the correlation between each two features and increase the accuracy of rating prediction. This happens because of the L2 regularization term in Ridge Regression. Ultimately, the advanced model 3 with the feature representation consisting of the intercept, the count of popular words, the length of review, the ordinally encoding price level, and the integer encoding review hour helped us in finding the best use of the existing features (by trying different types of encoding). And the added features of proportion of positive and negative sentences in the review text helped us in implementing our best model [Section 5.2.2].

7 CONCLUSION AND FUTURE WORK

In this report, we presented the user reviews rating prediction and analysis effort performed as our assignment 2, CSE258, Fall 2019, UCSD. Our main learnings are that the impact of text-derived information in predicting the rating of a review in a recommendation system is larger than just using some features like review length, time of posting the review. Another important aspect we have observed is that different categories might have different outcomes in terms of predictive model implementation. We have limited ourselves to a holistic analysis of all the reviews. We wish to explore different variants of the same dataset and implement models for personalized recommendations (KNN) and models which do not ignore temporal effects on recommendations as we learn from the remaining lectures. Rather than just using time as just a categorical feature, finding relations between the sequences of different places rated by users and predicting what the user will rate next would be the ultimatum for us as enrollees in this course.

8 REFERENCES

[1] Gayatree Ganu, Noemie Elhadadm, Amelie Marian, Beyond the Stars: Improving Rating Predictions using Review Text Content
[2] Yereya Berdugo, Review Rating Prediction: A Combined Approach
[3] Julian McAuley, Lecture 7 - CSE 258, Fall 2019, UCSD

[4] Ruining He, Wang-Cheng Kang, Julian McAuley, Translation-based Recommendation

[5] Julian McAuley, Lecture 9 - CSE 258, Fall 2019, UCSD