Predicting Ratings of Reviews using Temporal and Sentiment Analysis

ABSTRACT

This paper uses data from the Google Local Reviews dataset. Although the reviews from Google are preprocessed in JSON format, we still need to clean the text and time column. We used time features that we created by cleaning the given reviewTime column, including day of week, month, year, and Unix time. We also performed sentiment analysis by creating a bag of words and Term Frequency Document Frequency (TFIDF) and one hot encode them as features. Using these features, we trained a ridge linear regression model to predict the ratings a user would give to a particular location.

1 INTRODUCTION

Google is one of the most powerful technology companies in the world. Every day, 4.3 billion people use Google in their daily life. One feature that Google offers is Google Reviews, where Google users can leave reviews for all types of businesses and locations, including movie theaters, restaurants, landmarks, and even schools and police departments. All reviews have a rating ranging from 1 to 5 and the user who rated it. Most users include a text review in addition to the numerical rating. Many scholars have researched the impact of online reviews on retail performance and popularity.

In this research paper, we extracted time features and text features from Google Local Reviews data and tried to predict the rating using these features. Predicting the rating from the review text and comparing it against the actual rating can determine a discrepancy between whether or not the review was real or fake. Therefore, an effective model would give us vital information on the protection of a location's review against spam or bots.

2 DATASET FEATURES AND EXPLORATORY DATA ANALYSIS

The Google Local Reviews data consists of 8 features, which includes rating (int), reviewerName (string), reviewText (string), categories (list of strings), gPlusPlaceId (float), unixReviewTime (float), reviewTime (string), and gPlusUserId (float). We sampled 100,000 reviews from the dataset and split them into three parts: 80,000 for the training set, 10,000 for the validation set, and 10,000 for the testing set. Out of the 100,000 reviews, 30,813 are missing review text, 6,545 are missing categories, and 5,540 are missing Unix review time and review time.

Feature Variables Overview:

- reviewerName - name of the reviewer
- reviewText - customer review text
- categories - a list of categories of the place being reviewed
- gPlusPlaceId - id of the location being reviewed
- unixReviewTime - unix time the review was made
- reviewTime - time
- gPlusUserId - user id of the customer
- Day_of_week - day of week the review was made
- month - month the review was made
- year - year the review was made

Ratings range from 1 to 5, where 1 represents the lowest satisfaction and 5 represents the most satisfaction. The mean rating is 4.092 with a standard deviation of 1.189.

To summarize the review time column, we had to clean and then generate time features from the given reviewTime column. We converted the reviewTime column into a datetime object. We then extracted the day of week, month, and year from the datetime objects. Year takes values of 1990, 2003 to 2014. There are no years between 1991 and 2002. Week takes values from 0 to 6, where 0 is a Monday and 6 is a Sunday. Lastly, month takes values from 1 to 12.

The average rating of the sampled dataset fluctuated over time from 2000 to 2014. A likely reason for the variance is the fact that there were fewer data points in the early days of Google Reviews, leading to a larger variance before 2008. The average rating from 2008 and after stabilized with a smaller amount of variance due to more data points. Figure 1 graphs the relationship between average ratings over the years of the dataset.

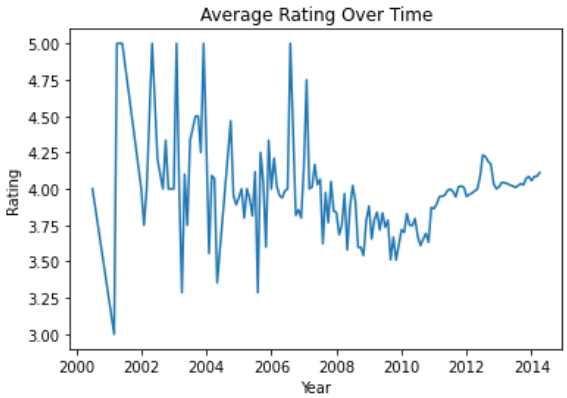

Figure 1: Average rating of all locations over time, binned by months.

We also graphed the cumulative proportion for ratings and found the data is skewed with more than 50% of ratings being five stars. Less than 25% of the ratings are 4 stars, 12.5% 3 stars, and so on.

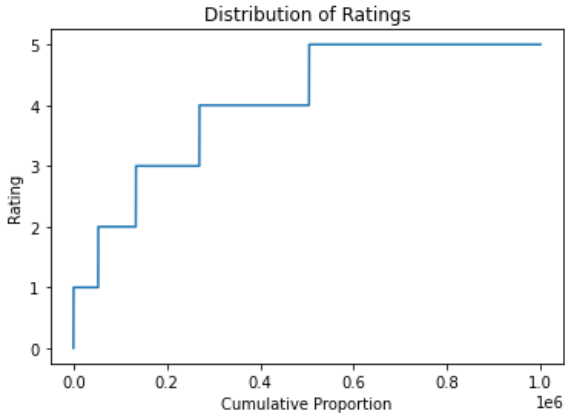

Figure 2: Cumulative distribution function of ratings.

Among the categories of restaurants, hotels, and other top categories, they all hold a similar distribution of ratings. This can be seen in Figure 3 and Figure 4. This holds true for most categories with a large enough sample size.

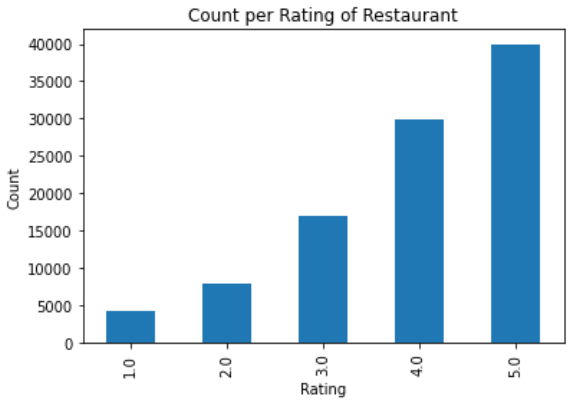

Figure 3: Rating counts for each rating value in the restaurant category.

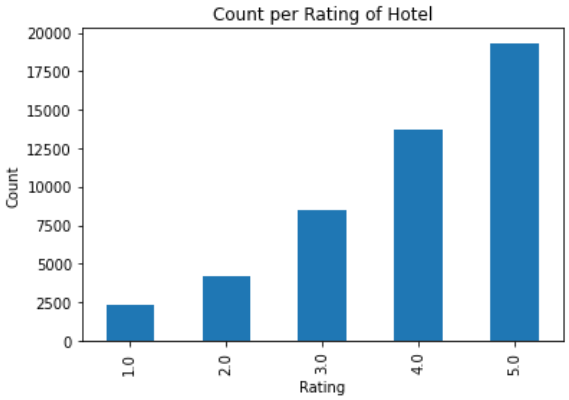

Figure 4: Rating counts for each rating value in the hotel category.

Within the top ten categories, we found the average ratings of restaurants within these categories are all less than about four stars. Most of the categories' average ratings are about the same with the exception of fast-food restaurants having the lowest average ratings at around 3.6 stars. This is likely due to how fast food's signal value is that of cheap and unhealthy food with newly trained first-time workers as staff. A reasonable assumption for why cafes hold the highest average ratings is likely due to how they are fancier and more expensive than most other places, assuring quality through higher pricing.

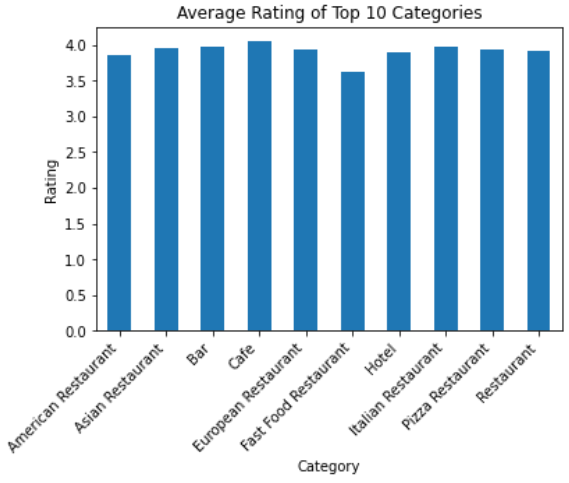

Figure 5: Average rating for the top 10 categories.

3 DATA CLEANING, MODEL EVALUATION, AND FEATURE SELECTION

We had to clean the data for further processing due to inconsistency in formatting and the number of null values. First, we converted the reviewTime column into a datetime object to standardize it. We then extracted the day of week, month, and year from the datetime objects. In total, we included 5 different types of temporal features: missing date, days of week, month, years, and Unix time. To handle missing review times, we set the feature vector to just zeros alongside the offset term. The review dataset included categories as a column, so a merge was done to join the categories to the locations dataset.

For model evaluation, we used MSE (mean squared error) as our error metric. We chose MSE over accuracy due to our outputs being a cardinal variable rather than a simple categorical variable. Using MSE, predicted ratings closer to the actual rating should be penalized less than a predicted rating that's far from the actual rating. However, if we used accuracy, any wrong prediction would be penalized the same regardless of how wrong it is. We can model MSE by:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - X_i\theta)^2$$

With mean squared error, we compute the mean of the actual star ratings minus the predicted ratings squared.

After exploring the data, we looked into what type of data we could feature engineer for use in our machine learning model. Different features we set up included a unigram and bigram bag of

words, TFIDF (term frequency inverse density frequency), one hot encoding of the 2000 most popular categories, alongside date-time features. With the unigram and bigram bag of words, we included the top 2000 combinations of unigrams and bigrams. We also compute the TFIDF of every word in each review using data from a unigram bag of words. Due to the formatting of the category column as a list, we had to extract each individual category from each location's list of categories. After, we one hot encoded for the 2000 most popular categories, holding different weights for each category in our regression model. A value of 0 in the feature vector means that the current review does not have one of the top 2000 categories as one of its categories, with 1 meaning that the category exists in the review and the top 2000 categories. With the date-time objects, we used one hot encoding on the 7 days of the week, 12 months, and 13 different years. We included a binary variable for if the date of the review is missing and used Unix time's value in seconds. To create the bag of words with both unigrams and bigrams, the data had to be preprocessed in multiple steps. First, we used word stemmer from the python NLTK (natural language toolkit) library to transform words such as "cars" to "car" and so on. We then removed punctuation and stop words, such as 'a', 'the', 'or', 'so', etc. from the review text. With this, we entered all the words in a words count dictionary which we eventually sorted and filtered to keep only the most used 2000 unigrams and bigrams. These top 2000 words became the bag of words.

For our baseline model, we predicted ratings in the test set by using the mean rating of the training set. The mean rating of the train set was 4.0412, so we created an array of 4.0412 with the size of the test set. The resulting MSE was 1.3957. We will be comparing this MSE with our final model to assess the magnitude of the improvement.

4 THE MODEL

We used ridge regression for our final model with alpha equals 10. As we explained in section two, we used 34 features to represent time, 2000 features to represent the top categories, 2000 features to represent the top bag of words, and 10000 features to represent the words with the highest TFIDF.

We tried out four different models to predict ratings based on review data, including linear regression, logistic regression, balanced logistic regression, and ridge regression. After running all four models side by side with default hyperparameters for each model, we found that the ridge regression has the lowest MSE of 1.102; the linear regression comes in a close second place with an MSE of 1.111; both logistic regressions models performed poorly, with MSE of over 2 for both models. Therefore, we selected the ridge regression model.

To further optimize the model, we ran a regularization pipeline on the ridge model. We tested six different alpha values: [0.01, 0.1, 1, 10, 100, 1000, 10000]. Using our validation set, we found an alpha of 10 yields the lowest validation error of 1.1266. Applying this model to the test set, we obtained a test MSE of 1.046. Thus, we selected our alpha as 10 for our model's hyperparameter.

When we took a look at the predictions coming out of the ridge model, we found something interesting. The maximum value of the list of predictions was 7.739, and the minimum value was -3.665. Since we only have ratings ranging from 1 to 5 inclusive, any value above 5 or below 1 doesn't make sense. So, we wrote a function that converts values above 5 to 5 and

values below 1 to 1. By applying this function to the ridge prediction, we decreased our MSE from 1.046 to 1.015.

In addition to the unrealistic predictions discussed in the previous paragraph, we also ran into underfitting problems in the beginning. Initially, we did not include the TFIDF features; we used 34 features to represent time, 5,000 features to represent the top categories, and 5,000 features to represent the top bag of words. With only these types of features, we obtained an MSE of 1.015, which is much higher than the MSE of our final model of 0.915.

Compared to linear regression, ridge regression is an extension of linear regression. It was modified to minimize the complexity of the model. This modification is done by adding a penalty parameter (a regularization term) that is equivalent to the square of the magnitude of the coefficients. With the regularization modification, our ridge regression model performed better than linear regression in this case. On the downside, ridge regression requires more computing power to fit due to the addition of the penalty term.

Both the ridge and linear regression performed much better than the logistic models. Since the logistic models can only predict 1, 2, 3, 4, and 5, the minimum loss for a false prediction is 1 as opposed to less than 1 in the linear models. Since we chose mean square error as our metric to evaluate the models, logistic regression was not a good model choice. However, if we have chosen accuracy, logistic regression may have performed better.

5 RELATED WORK

Our dataset was built from [1] and [2] and was used for sequentially recommending locations based on a user's historical interactions. The authors created the state-of-the-art model *TransFM* which combined translation and metric-based approaches for recommending locations using Factorization Machines (FMs). They used temporal features, item category features, user and item content features, and geographical features. Time of review was used as a temporal feature, as well as standardizing the timestamp. Categorical labels were one hot encoded as binary variables for features as well. Data on the user was also used, such as the user's age, gender, occupation, and zip code for personalization of recommendations. Lastly, geographical data was used to recommend local locations.

[3] sequentially recommends locations based on post history as well, but approaches the problem with a different approach. [3] used a PRME model (Personalized Ranking Metric Embedding). This incorporated different features such as sequential information, individual preference, and geographical influence. They found that more than 50% of consecutive check-ins occurred less than 24 hours after the processive check-in. One geographic finding was that 70% of consecutive check-ins were within 10km of one another.

In [4], sentiment analysis from the text of reviews is also used to predict ratings. The paper does most of the same preprocessing on their data as us, but also expanded contractions such as haven't to have not as well as using negation handling before they removed stop words. This compounded words like "not worth" into "not_worth". [4] also made a feature for the polarity of a review using the adjectives in the cleaned text as well as the length of the review text. The paper decided to go with a classification model and used Multinomial Naive Bayes, Logistic Regression, and Linear SVC (Support Vector Classifier), achieving the

highest accuracy with Logistic Regression at 54.1%.

Monolith is the most recent and most state-of-the-art product scale recommendation system. The newest recommendation system from [5], authored by TikTok's parent company, ByteDance, utilizes deep learning in an industrial-level setting. It models its general architecture after the Worker-ParameterServer distributed system from TensorFlow, utilizing dense features in the neural network and sparse features in embedding tables. For the dense set of parameters, feature variables were efficiently stored inside a collisionless HashTable, allowing for online scalability and real-time updates to their recommendation system. They train the model within two separate stages. The first stage is the Batch training stage, which trains itself from historical data, which is useful when there is a modification done to the model architecture or when the model is retrained. The second stage is the online training stage, using the most recent history as training data to retrain the model, immediately taking an effect on the user side. This allows for feedback from the user in real-time.

6 RESULTS AND CONCLUSIONS

Our 100,000 review sample was split in an 80/10/10 ratio for training, validation, and testing, respectively.

Testing against linear regression, logistic regression, balanced logistic regression, and ridge regression, after testing each hyperparameter within each model, we found that ridge regression had an MSE of 1.102, linear regression with an MSE of 1.111, and both the logistic regression models performing poorly with MSEs over 2. We selected ridge regression as our final model.

With ridge regression and our alpha equal to 10 and flooring values greater than 5 to 5 and ceiling values less than 1 to 1, the final mean squared error was brought down to 0.915.

Making rating predictions based on the review text and comparing it against the actual ratings can determine if there is a discrepancy between the text and the rating. This will allow for the reviews of locations to detect whether or not a review is real and is to be marked as spam. With this, a system can be built to protect a location's reviews against bots.

REFERENCES

[1] Rajiv Pasricha, Julian McAuley, Translation-based factorization machines for sequential recommendation, *RecSys*, 2018

[2] Ruining He, Wang-Cheng Kang, Julian McAuley, Translation-based recommendation, *RecSys*, 2017

[3] Shanshan Feng, Xutao Li, Yifeng Zeng, Gao Cong, Yeow Meng Chee, Quan Yuan, Personalized Ranking Metric Embedding for Next new POI Recommendation, *IJCAI,* 2015

[4] Ankit Taparia, Tanmay Bagla, Sentiment Analysis: Predicting Reviews' Ratings using Online Customer Reviews, *SSRN*, 2020

[5] Zhuoran Liu, Leqi Zou, Xuan Zou, Caihua Wang, Biao Zhang, Da Tang, Bolin Zhu, Yijie Zhu, Peng Wu, Ke Wang, Youlong Cheng, Monolith: Real Time Recommendation System with Collisionless Embedding Table, *RecSys,* 2022