Predicting User Ratings for Businesses Using Google Local Data

Andrew Mokhtarzadeh
anmokhta@ucsd.edu

David Davila
ddavilag@ucsd.edu

Maricela Vasquez
mgvasque@ucsd.edu

1. Data

The data we are using for this assignment is a dataset from Google Local which contains reviews of businesses in California. For this project, we analyzed a subset of the entire dataset such that each user has reviewed a minimum of 5 businesses, and each business has a minimum of 5 reviews. Each review contains user_id, business_id, temporal data regarding when the review was submitted, the reviewed business's categories, the full text of the review, and the overall rating of the business on a scale from zero to five. The fundamental statistics for a few key values can be found in Table 1 and Table 2. The distribution of ratings is shown in Graph 1, where a four (out of five) is the median rating, and the mean and standard deviation is 4.05 and 0.88, respectively. In addition to the reviews dataset, we have a dataset with information for each California business on Google Local. This allows for more personalized data for each business, giving us insight into data such as general price point, address, hours of operation, and longitudinal and latitudinal data. Merging the data together gives us access and the ability to more easily explore the data and create meaningful features.

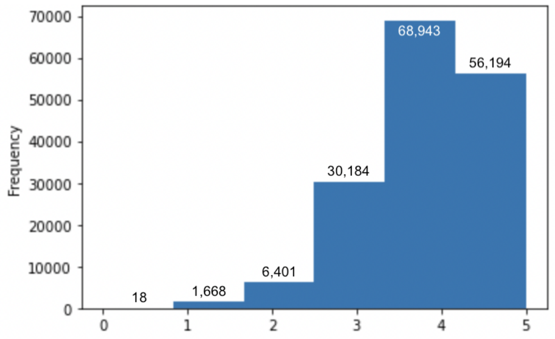
Table 1: Basic Data Statistics

# Users	# Businesses	# Categories	# Reviews
9,179	10,477	1,011	163,408

Table 2: Average Data Statistics

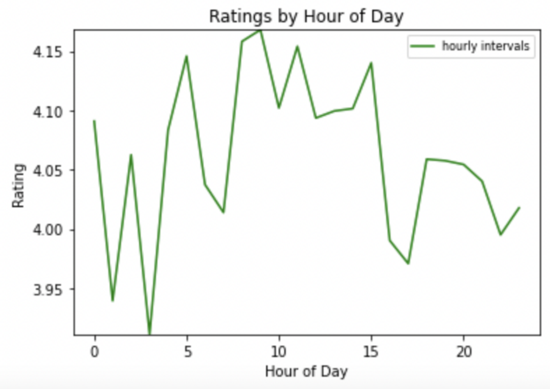
Avg # Reviews / User	Avg # Reviews / Business
~17.80	~15.60

Graph 1: Distribution of Ratings



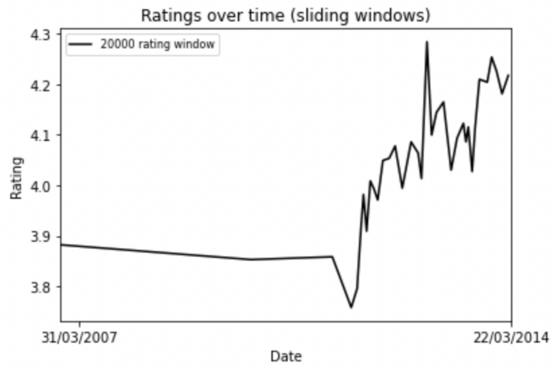
For our first task for exploratory data analysis, we decided to look at the distribution of reviews by the time of day they were posted. The first step was to convert the unixReviewTime, or the time reviews where posted, to datetime type in order to proceed with the distribution. With all of the time values converted into datetime in a list called ratingTime, we are now able to graph the distribution. Making the bins in hour intervals, we get distribution in graph two. The trend has a peak in the late morning. Most people would be writing reviews in the middle of the day and less likely in the middle of the night, so the trend in graph 2 follows this logic. However, it is clear from the graph that the average rating for each hour interval does not fluctuate very much: it stays within the range (3.95-4.15 stars).

Graph 2: Ratings by Hour of Day



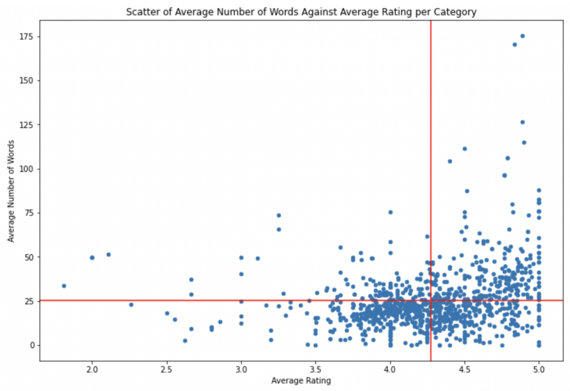
The next exploratory data analysis task we completed was to observe the distribution of reviews across date. With our reviewTime list from the previous analysis, the date was obtained by extracting the date attributes from our list containing datetime values of our reviews. In graph 3, we notice a positive trend, which suggests that ratings increase with the date.

Graph 3: Ratings Over Time



Our next exploratory task was looking at the relationship between the number of words in a review and the rating the review received. To visualize this relationship, we created a scatter plot and added the mean of both the x and y axis. Graph 4 has a cluster that surrounds the review rating and the average number of words in a review.

Graph 4: Average # Words by Average Rating For Each Category



For our last exploratory data analysis task, we observe the top ten words from the reviews text. Using a bag of words method to gather the word counts from all the reviews. Removing punctuation and stop words so that they would not interfere with our results. The final results of this are as follow:

```
[(28827, 'food'),
 (28762, 'good'),
 (27991, 'great'),
 (23042, 'place'),
 (14069, 'service'),
 (11499, 'like'),
 (11397, 'get'),
 (10658, 'best'),
 (10246, 'one'),
 (10117, 'really')].
```

Most of the top words such as ‘good’, ‘great’, and ‘best’ are seen to have a positive connotation, resulting in a better rating. Having more positive words than negative words might suggest that reviews that appear on the dataset are all that give high ratings, which is seen in our average rating of 4.09.

2. Predictive Task

Our assignment’s predictive task is to predict what a given user would rate a given business on a scale of one to five.

2.1 Features

In order to properly predict what any given user would rate a business, a series of features must be extracted and created from the data given. We obtained the following features in order to create our baseline models, evaluate our main model and create accurate predictions:

Temporal Dynamics: After viewing an upward trend in ratings as shown by the Sliding Window graph of reviews over time, we implemented a simple regression-based approach to capture temporal trends. This implementation was copied from Chapter 7 Workbook (Exercise 7.4). Every review in the training dataset is stored in a defaultdict(list) with the business_id as the key and (time, user_id, and rating) as the values. For a given (time, user_id, business_id), this feature predicts the rating based on the 5 most recent previous ratings from the business, and the average of all ratings from the business.

User Rating from Cosine Similarities: Looking at every other business a user has reviewed (in the training dataset), we are able to compare these new businesses against the given business we are trying to predict a rating for by finding the cosine similarity between businesses. The similarity can be calculated using the definition below:

$$cos(\theta) = \frac{A \bullet B}{||A|| \times ||B||}$$

Where A is a list of all reviews given for the business we are trying to predict the rating of and B is a list of all reviews for a business that the given user has reviewed before.

Business Pricing: Each business is categorized based on how expensive it is, ranging from expensive ("$$$") to cheap ("$") to no prices associated (""). Missing and blank values were grouped together before converting the category labels to binary features through One Hot Encoding. The idea behind this feature is that we expect reviews to be more critical for more expensive businesses.

Review Text Sentiment Analysis: For this feature, we extracted the 1000 most popular n-grams up to length 5, not including stop words, from every review in the training dataset and added them to a dictionary. Then for each prediction we made a feature vector of length 1000 which one-hot encoded whether or not one of the 1000 most popular n-grams (upto 5) was present or not in the review text. Used implementation from Chapter 8 Workbook.

User Rating from Item2Vec: For this feature, we created a defaultdict with user_id as the key and (time, business_id) as the values for every review in the training set. From this dictionary we initialized a item2vec model from the gensim Python module which estimates the probability/similarity that businesses appear nearby. We then used the heuristic from Chapter 8 Exercise 8.5 in which we predicted the rating from the item2vec similarity scores. This heuristic works by taking the difference of all ratings from the user from each business's respective rating average before.

2.2 Evaluation Methodology

In order to properly evaluate how well our model's predictions perform when compared to their true rating counterparts, we will report the Mean Squared Error, or MSE, as defined below:

$$\text{MSE} = \frac{1}{n}\sum_{i=i}^{n}(Y_i - \widehat{Y}_i)^2$$

where Y_i is the true rating given and $\widehat{Y}_i$ is the estimated rating. Additionally, using SKLearn we partitioned our reviews dataset with a 70%/20%/10% train/validation/test split in order to properly test and fine-tune hyperparameters in our model.

2.3 Baselines

We compare our main model against a series of predetermined baselines:

Baseline 1 (Global Average): A naïve baseline that returns the global average rating of the training data, regardless of any feature. It is not personalized and always returns the same value. The observed MSE for this model on the test set was 0.764.

Baseline 2 (Weighted Category Average): For this baseline model, each prediction was calculated by determining the average rating for each category a business is labeled under and computing the average between all the category means. The observed MSE for this model on the test set was 0.727.

Baseline 3 (User Rating Based on Cosine Similarity): This model computes the cosine similarity between two businesses just like the feature as described in Section 2.1. The observed MSE for this model on the test set was 0.718. This was the best baseline.

3. Model

The model we decided to use was a Ridge Regression Model with five features: sentiment analysis using the 1000 most popular n-grams upto length 5, one-hot encoded price, item2vec, user rating from cosine similarity, and temporal. For predicted ratings > 5, we set these equal to 5; similarly, for predicted ratings < 0, we set these equal to zero. We noted that the MSE of this model on the training set was 0.454 and the MSE of the validation set was 0.607. We attempted to use a variety of other models to predict our ratings to ensure we were using the best predictor. For example, we attempted to use a Decision Tree Regression model with the same five features above, but found that the MSE of the model on the training set was 0.580 and the MSE of the model on the validation set was 0.640, significantly worse than the Ridge Regression model we decided to go with. Using a Random Forest Regression model produced similar MSE's that were also significantly higher than the Ridge Regression model as well.

4. Related Literature

The google local business dataset was originally used in the Translation-based Factorization Machines for Sequential Recommendation. In this report, the features that were used for their models included temporal, item-category, user and item content features. The models covered in the report included BPR-MF and personalized ranking metric embedding, and Hierarchical Representation. The BPR-MF did better than the popularity-based baseline. The personalized ranking metric embedding model takes the euclidean distance between two features and replaces the inner product. The final model used on the dataset was the Hierarchical Representation Model. This model takes the first term of the inner product, then takes the sum of both embeddings as demonstrated in the equation below:

$$\hat{y}(\vec{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \vec{v}_i + \vec{v}_j, \vec{v}_i \rangle x_i x_j .$$

The experiment in the report primarily focuses on the comparison between the Personalized Ranking Metric Embedding and Hierarchical Representation models. The results revealed that the Hierarchical Representation was better on the google local business dataset than Personalized Ranking Metric Embedding [1]. The results from this report were similar to our model because we also found that the simpler model performed better. We found that removing a few features had better outcomes.

The dataset can also be used in Translation-Based Recommendation to make recommendations. The recommender in the report used Weighted Nearest Neighbor, Low-Rank Mahalanobis Transform. The Weighted Nearest Neighbor model measures the dissimilarity between two items by Euclidean distance. The Low-Rank Mahalanobis Transform is an embedding method that learns the compatibilities between two items [2].

5. Results

Ridge Regression Model: We were most pleased with the overall performance of the Ridge Regression model in comparison to the 3 Baseline models and alternative regression models in terms of MSE: 0.607 for this model compared to 0.764, 0.727, and 0.718 for the respective baselines.

Ablation Experiment: However, the large discrepancy in MSE between the train and validation sets led us to believe that this model was overfitting the training data. As such, we conducted an ablation experiment in which we removed each feature from the model one-by-one. For our final model, we ended up completely removing both the user rating from cosine similarity and temporal regression features due to the decreases in validation MSE when these features were removed during the ablation experiment. As such, the train and validation MSE's for the final model were 0.510 and 0.575, respectively. Since these two MSE values are much closer, this suggests that our model is not overfitting the training data.

Feature Importances: From the results of the ablation experiment, the ranking of features from most-least important was: review text sentiment analysis using n-grams upto 5, Item2Vec, One-Hot Encoded Price, Temporal Dynamics, and the User Reviews from Cosine Similarities of Businesses. The Validation MSE's from each step of the ablation experiment were 0.691, 0.611, 0.609, 0.599, and 0.573 for the respective features above. Clearly, the Sentiment Analysis Feature using the 1000 most popular n-grams of review text data proved to be the most important feature since the Validation MSE increased drastically when it was removed from the model.

Hyperparameter Tuning: Once we finalized which parameters to use, we performed hyperparameter tuning of the final Ridge Regression model using the validation set, and determined that the regularization coefficient C=1 gave the lowest MSE values on the validation set.

Reported Test MSE: For our final step, we calculated the MSE to be 0.568 from the predictions of this tuned Ridge Regression Model on the untouched test set. This means that on average, this model predicts ratings 0.753 [or sqrt(0.568)] stars away from the true rating for this dataset.

References

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

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

[3] Farman Ullah, Ghulam Sarwar, Sung Chang Lee, Yun Kyung Park, Kyeong Deok Moon, and Jin Tae Kim. 2012. Hybrid recommender system with temporal information. In ICOIN. IEEE.

[4] Yehuda Koren. 2010. Collaborative Filtering with Temporal Dynamics. Commun. ACM (2010).