Rating Prediction & Sentiment Based Recommendations for Google Maps Businesses

Hargen Zheng
Halıcıoğlu Data Science Institute and Dept of Mathematics
University of California, San Diego
La Jolla, California, USA
hgzheng@ucsd.edu

Nathaniel del Rosario
Halıcıoğlu Data Science Institute
University of California, San Diego
La Jolla, California, USA
nadelrosario@ucsd.edu

Chuong Nguyen
Dept of Computer Science and Engineering
University of California, San Diego
La Jolla, California, USA
chn021@ucsd.edu

Trevan Nguyen
Halıcıoğlu Data Science Institute
University of California, San Diego
La Jolla, California, USA
ttn077@ucsd.edu

1 INTRODUCTION

Your anniversary with a loved one is coming up, but you need help figuring out where to dine. Or you're looking for a nearby gym with an affordable price to keep your body in shape but don't know where to look. Often, we find ourselves stuck between choices and are afraid to make one simply because we want to make everyone happy. Yet, as college students, our money pockets are too shallow to fulfill all the choices. So, the worst of the worst, we fall victim to our choices and are the scapegoats for our loved ones' happiness. In this project paper, we intend to resolve the decision-making problem by offering recommendations of popular places with various customer service types. To build a robust recommender system, we have used California's Google Local Reviews dataset, consisting of a Business dataset (515,961 rows) and a Reviews dataset (44,476,890 rows). For the simplicity of this project, we will mainly use review text as our training feature and the star rating as our label. Since the star ratings in our current dataset are very skewed and nearly perfect, we have converted these star ratings to binary using a tuned threshold. This means a star rating above a certain tuned threshold will be marked as one or, in other words, is to be recommended. The dataset is very large; thus, we can't recommend a business out of the entire pool of California; however, we can still make k recommendations from a large sample. As we want to recommend places that a user has a high possibility of going to, we will only pick the top k recommendations, meaning places with the highest star rating will be recommended. This approach has yielded many limitations which we will dive deeper into in the result section.

Decision-making can sometimes be challenging, especially when treating your loved ones to a fine dining restaurant, workout places, etc. on a special occasion; such feelings may discourage us from trying new places. Since we do not want to embarrass ourselves or disappoint our loved ones, we would instead stick to much safer and boring options. However, our lives don't have to be that depressing. This paper will introduce a recommender system that can turn embarrassment into good impressions. If you are studious but want to invite your crush to a high-quality restaurant, we have your back. Being a Triton does not mean we can't be trendy. Being a programmer does not mean we always live in the dungeon and have no social life.

1.1 Dataset Description

To build a robust machine learning model, we would need a large enough dataset to provide large amount of training data – so our model can learn the well enough to make reasonable predictions. Therefore, we chose the Google local review dataset, which contains both business and user reviews. Since it contains data for the entire nation and would be too large for local computers to process, we decided to focus specifically on the subset of the dataset that contains California data. Therefore, our recommender system prediction will not guarantee total generalization outside of California since certain companies may not exist. After collecting the subset of California data, we have $1,007,571$ rows of business reviews. Out of these review data, $6,344$ do not have record for rating and $446,094$ do not have text review record, each constituting 0.6296% and 44.274% of the original subset of California data. With the available rating data we have, the average rating is roughly 4.2094, which seems high. Therefore, we analyzed the distribution of review ratings and the result is shown in the barplot below:

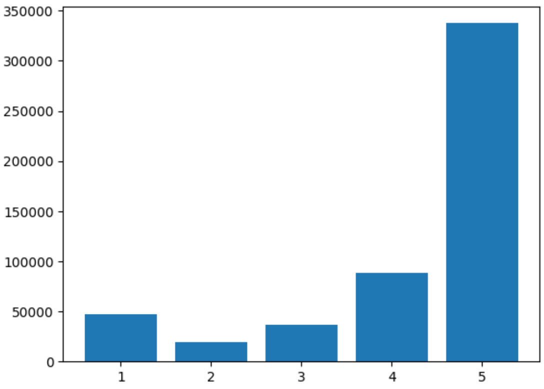


Figure 1: Review Rating Distribution

Indeed we have an unbalanced rating distribution, which means the reviewers in our dataset tend to be generous with the rating they give to businesses. In such a case, we believe ratings of 3 or below would be on the lower end of user ratings and thus are more likely to have negative reviews. To backup our assumption, we took a deeper look into the dataset and found the following samples of text reviews, where their corresponding rating follows in the parentheses.

- "Their middle school age customers are rude and the staff don't care to stop them." (3)
- "Not happy. Overcharged. Did extra service without consulting me." (3)
- "Convenient location, but usually busy. Staff is hit or miss, and the prices are usually a little higher here than other parts of Lancaster." (3)
- "I think what they're doing for the homeless is awesome." (4)
- "I like this doc! I had a great first encounter. However, I was probably having unrealistic expectations from the system and got a bit frustrated. Overall, the doc was open and sincere." (4)
- "I came in today not knowing what kind of hairstyle or color I wanted to get, however the stylist Joni was so sweet and helped me figure out what style and color would be best for me." (5)

This shows that ratings below 4 would generally come with more negative text reviews, as expected.

2 PREDICTIVE TASK: N-GRAMS MODEL BASED RATING PREDICTION

Given our current dataset, we will implement the Bag-of-Words/N-Grams technique to perform Sentiment Analysis. We chose to do Natural Language Processing because we intended our recommender system to predict ratings, which correlates highly with review comments. To evaluate our model, we will use the average star ratings from the dataset, which has already been converted into binary through fine-tuned threshold comparison. We will then compare these labeled ratings to our predictions/rating recommendations and calculate our accuracy rate by summing up all of the correct predictions and dividing the value by the total predictions.

For data processing, we removed NaN entries for rating and text. We also ignored ratings with less than a length of 10 because the 95th percentile of review length has the length exactly 10. The reviews of the length less than 10 likely do not contain meaningful data. There were $6,344$ NaN ratings and $446,094$ NaN texts. Less than 1 percent of the ratings were NaN and about 44 percent of the texts were NaN. This is likely because many people rate the place without leaving a review.

For the model, we will be using N-grams in which the following section will dive into which N in N-grams will yield us the most optimal result. Additionally, to improve the performance, we have decided to combine multiple N-grams models with each n to be different from one another. The reason that chose N-Grams was because single unigram words extraction for Sentiment Analysis will not work optimally. Since the dataset that we are using are entirely in English, and lots of English words have multiple meanings depending on the context. Hence, to avoid word meaning confusion when doing Sentiment Analysis, we will use N-Grams to boost context. What N-Grams Model does is essentially taking in not just the current word but also previous $N-1$ words in the sentence. Despite N-Grams allows us to incorporate context, but the usage of previous words will greatly reduce our dictionary of words as we can't include starting $N-1$ words due to out bound reason. To offset the limitation, we will ensemble multiple N-gram models of different N and show how this approach will give us better results in the following section.

For the data labelling, we analyzed the class distribution and examples discussed in part one and chose a threshold for the positive and negative label split because even though the mean rating of the data was skewed extremely right, the actual text responses showed that reviews of 3 and lower were where the negative reviews were. Thus we concluded that people from this dataset have a tendency to positively review a business.

As suggested by the barplot in the previous section, the positive and negative labels are pretty imbalanced, where positive labels has a much larger count than the negative labels. Therefore, we sampled from the original 500,000+ rows and only sampled up until the point that the two classes – positive (above 3) and negative (3 and below) – were of equal size, which were $107,313$ for each class. This is the main uncertainty with our model; with a class imbalance this big, it is possible it will not generalize well being only trained on a subset of the entire dataset. Other approaches that could be taken are taking a non-text based approach or using a classifier that allows for class weighting such as logistic regression and boosted decision trees to get a better generalizing model.

To evaluate the model performance, we developed a baseline model, which just predicts the majority label in the dataset. The baseline model has about 50.5% accuracy on a $107,313$ and $107,313$ split of positive and negative labels.

To achieve an improvement on our baseline, which was just voting the majority label, we tried out numerous N-Gram models. To evaluate our model, we used a standard accuarcy measure of number of correct predictions divided by the total size of the dataset/validation set.

Our model design was relatively simple. The two relevant features for our model were the average business rating (quantitative discrete) and the review text string. Additionally, the preprocessing was pretty straightforward. Since the data was pretty structured, we were able to read in the lines and use Python dictionaries to store the data in an easily accessible format, which would end up helping speed things up with pre-computations.

3 N-GRAMS MODEL

Since we are working on predicting the rating based off the text reviews, we would need to capture how the appearance of each word in the dictionary, obtained by extracting words from text reviews in the dataset, correlates to the rating that a given user gives to the business. Therefore, we decided to implement sentiment analysis based on N-gram model.

3.1 Unigrams Model

In this simple model, we can estimate the parameter values from the training data as follows:

$$P(w) = \frac{\text{Count}(w)}{\text{Total number of words}}.$$

We started with the native unigram model shown above. The implemented model for rating prediction is a linear regression model with ridge regularization. The purpose of regression was to predict the sentiment of the text data based on the rating feature. The softmax equation is provided as follows:

$$P(y = i|\mathbf{x}) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}},$$

where score z is in $z = [z_{1:k}]$.

We then used a softmax to convert every predicted sentiment into a class label. Any review with a evenatual score above 3 would be labeled as a positive review and any review with a score equal or less than 3 would be classified as negative review. This choice was motivated by the model's simplicity and interpretability, providing a baseline for comparision. The features utilized in the model are unigrams, represented through a bag-of-words approach and the target variable is the ratings extracted from the review dataset. Ridge regularization was incorporated to prevent overfitting by penalizing large coefficients, particularly essential in high-dimensional datasets.

3.2 Bigrams Model

After implementing the basic Unigram model, we implemented the Bigrams model for the sake of model comparison. This was a continuation of the base Unigram model, following the standard formula for the Bigrams model as follows:

$$P(w_n|w_{n-1}) = \frac{\text{Count}(w_{n-1}, w_n)}{\text{Count}(w_{n-1})}.$$

This model performs slightly worse than the unigram model. We followed the same ridge regression and softmax pipeline for all the N-gram models that follows the current section.

3.3 Trigrams Model

Similar to Bigrams model, we also initially implemented a Trigrams model for the sentiment analysis task. The model is given as follows:

$$P(w_n|w_{n-2}, w_{n-1}) = \frac{\text{Count}(w_{n-2}, w_{n-1}, w_n)}{\text{Count}(w_{n-2}, w_{n-1})}.$$

When training the Trigrams model, we expect that as the more words are taken into consideration, the more accurate our sentiment estimation and therefore rating prediction would be. However the performance for our first run of this model was worse than the previous two models. Along with the performance of Bigrams model, this makes sense because as we increase N for N-grams, the lower level representations of the text data are increasingly missing. Therefore, it becomes harder for the model to capture detailed correlation between individual words and ratings.

3.4 5-grams Model

The model is given below:

$$P(w_n|w_{n-4}, w_{n-3}, w_{n-2}, w_{n-1}) = \frac{\text{Count}(w_{n-4}, w_{n-3}, w_{n-2}, w_{n-1}, w_n)}{\text{Count}(w_{n-4}, w_{n-3}, w_{n-2}, w_{n-1})}$$

The 5-grams model is the worst among all the models we have performed – this matches our finding when performing the Trigrams model.

3.5 N-grams Model Ensemble

Up until this point, we have finished performing all basic models. We decided to ensemble individual N-gram models together to obtain a more accurate and robust model to perform the sentiment of text reviews. Also, since we are incorporating two separate models together, we anticipate the model to generalizes better than the individual N-gram models which were trained on 90/10 splits initially. This makes sense conceptually as we originally hypothesized that taking more preceding words into account will give a better performance. One shortcoming of these findings is that we would need to do more cross validation among different data splits on a larger dataset to get a better idea of the generalization capabilities, which was not within the capacities of the given time and computing power.

3.6 Unigram and Bigrams Ensemble

To our surprise, Unigram and Bigram model worked well individually, so we hypothesized that if we combined them, the performance may also improve. Our approach was as follows:

$$P(w_n|w_{n-1}) = \lambda P(w_n) + (1 - \lambda)P(w_n|w_{n-1})$$

Here, lambda acts as a weighting parameter to favor either Unigram or Bigram models, which we ended up spending a lot of our efforts on tuning.

We also performed various other ensembles of N-gram models, such as Unigram, Bigrams, Trigrams ensemble, Unigram, Bigrams, Trigrams, 4-Grams ensemble, etc. Out of all these ensemble models, the Unigram and Bigrams ensemble outperformed the others.

3.7 Fine Tuning

The optimization process involved fine-tuning the regularization parameter lambda and considering feature engineering parameters, such as top K words that we consider when constructing the feature vector and regularization constant λ. The tuning process is shown in the results section.

3.8 Model Evaluation: Strengths and Weaknesses

The strengths of the chosen model lie in its simplicity, interpretability, and regularization mechanism that prevents overfitting. However, its weaknesses include the assumption of a linear relationship between features and the target variable, which may not capture intricate patterns. The model's limited expressive power compared to more advanced counterparts and its sensitivity to outliers are also noteworthy.

The selected model model establishes a baseline for rating prediction using unigram features and linear regression with ridge regularization. Its performance will be assessed against more complex models to determine whether increased complexity justifies improved predictive accuracy. The fine-tuning of the regularization parameter and feature engineering parameters should be informed by cross-validation results.

Several challenges were encountered during the implementation. Scalability became an issue depending on the dataset size and feature space, while overfitting was mitigated through ridge regularization. The comparison models considered considered encompassed variations in the feature representation, regularization methods, and non-linear models.

Unsuccessful attempts included attempts included experimenting with different feature representations, which did not consistently lead to performance improvements. The exploration of non-linear models without appropriate feature engineering also posed challenges.

4 LITERATURE REVIEW

We used the Google Local Dataset from 2021. The datasets are personalized datatset built by Professor Mcauley's group for the following research papers [3] [5] [4] and [2]. Generally, a robust recommender system requires a very large dataset, hence we combined the two dataset to build ourselves a large dictionary of words. This is significance because we are using Sentiment Analysis, thus, reducing ourselves to single training feature which is not an ideal scenario given that we want to build a recommender system.

An approach we were considering was using sentiment polarity categorization. The goal is to classify text into positive, negative, or neutral sentiment. The article referenced involving creating lists of positive and negative words based on customer reviews [1]. These lists of words also included misspelled words which is common in social media and review content. Sentiment categorization is treated as a classification problem, where the identification of features containing sentiment information is crucial. The article also proposed a text categorization technique that removes objective sentences by extracting subject ones using a minimum cut approach. In this case, the researchers used Twitter data, selecting 6,799 tokens and each token was assigned a sentiment score called the TSI (Total Sentiment Index), which is computed as the ratio of total number of positive tweets to the total number of negative tweets in which the token appeared.

This was a potential route we considered taking our project towards but considering the Google Local Dataset, it seemed to be highly skewed toward positive sentiment and ratings, which would potentially skew the analysis and results.

5 RESULTS

Despite our current baseline model, which will work with recommending top-rated places, the model that we have and the approach that we are taking has lots of limitations. First, since we are using a collective dataset from Google, there are many different businesses, such as restaurants, gyms, shops, etc., which is a big problem. For example, a user x wants our recommender system to recommend a top-rated restaurant for a birthday party. Still, since the recommender system offers top-rated places without considering the type of places the model suggests, the recommender system can recommend the user x a gym place instead, which needs to be corrected. Additionally, since we are using Sentiment Analysis to make predictions on ratings of places based on comment reviews and our model only recommends a fixed number of top places in the order of ranking, this approach would fail to make personalized ranking. This is because the model only considers k top rated places, thus, the recommendation list will be the same for all users. Which is very bad because not all users have the same taste, like the same food. Additionally, assuming that our model continues to use the same training set for the recommendation, hence, the model will essentially be recommending the same k items to users from day to day, which is not ideal as a recommender system. This may also leave places with a rating slightly lower than the k top rated places to be misrepresented.

Finally, the performance results of our model are described in the following subsections.

5.1 Model Performances

As noted in previous sections, Unigram performs the best out of individual models without ensemble, and Unigram and Bigrams ensemble works the best for ensemble technique. Before tuning parameters, we only considered all text reviews that are not None, considered Top 3000, and set Ridge Regression regularization constant to be 1. The following table shows model performances on validation dataset (10% of all data we consider), with the above hyperparameter values.

Table 1: Different Model Performances

Model Type	Validation Accuracy	Split
Unigram	0.78372	90/10
Bigram	0.71919	90/10
Trigram	0.64236	90/10
5-gram	0.77000	90/10
Unigram Bigram ensemble	0.78376	90/10

5.2 Hyperparameter Tuning

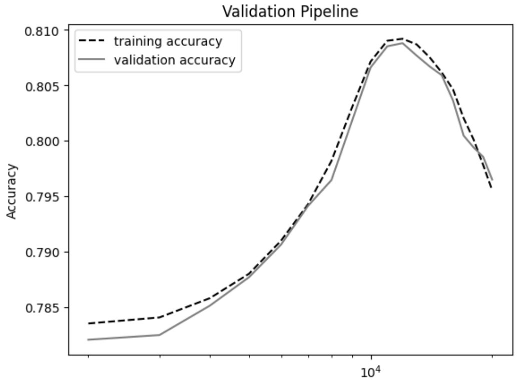
After comparing model performances, we started fine-tune hyperparameters on Unigram and Bigrams ensemble model to improve its performance. With the current computing power and time restriction, we tested over 20 values of lambda and 4 values of k

Firstly, we have tuned the regularization constant. We initially attempted the λ values from 0.01, 0.1, 1, 10, 100, $1K$, $10K$ and realized that as the regularization constant increases, our model performs better and better. Therefore, we then tried values from $2K$ to $20K$ and the result is shown below:

Table 2: Tuning of Lambda

Lambda	Validation Accuracy	Lambda	Validation Accuracy
2000	0.782053671263511	3000	0.7824729780096906
4000	0.7851285874021617	5000	0.7876910175177041
6000	0.7906261647409616	7000	0.7941203876257921
8000	0.7964498695490123	9000	0.8018076779724189
10000	0.8066064107342527	11000	0.8085165859112933
12000	0.8087961237420798	13000	0.807677972418934
14000	0.8066995900111815	15000	0.8059075661572866
20000	0.7964964591874767		

The corresponding accuracy plot is shown below:

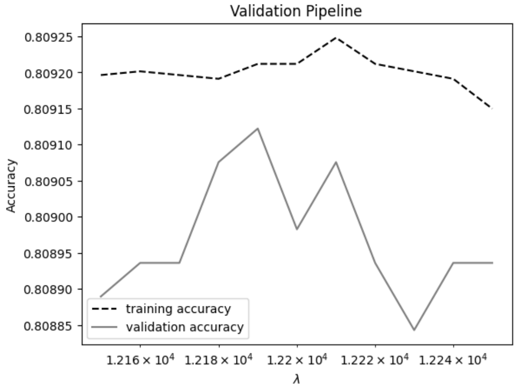


We realized that our model performs the best at around $12K$. Therefore, we further tried out λ values from $11.5K$ to $12.5K$. The result is shown below:

Table 3: Fine Tuning of Lambda

Lambda	Validation Accuracy	Lambda	Validation Accuracy
12150	0.8088893030190085	12160	0.808935892657473
12170	0.808935892657473	12180	0.8090756615728661
12190	0.8091222512113306	12200	0.8089824822959374
12210	0.8090756615728661	12220	0.808935892657473
12230	0.8088427133805441	12240	0.808935892657473
12250	0.808935892657473		

The corrsponding accuracy plot is as follows:



With this iteration of fine-tuning λ, we identified that the best λ value would be 12190. The regularization constant is pretty large, meaning the coefficients of each word in our feature vector are pretty small. It's reasonable because we have $3K + 1$ elements (including the offset term) to represent each of the text review. If the coefficients are large, the final results would be strongly dominated by more frequent words in the feature vector. By making coefficients small, the resulting rating predictions would then be more evenly contributed by each element in the feature vector, thus better capturing the entire sentiment.

Then, we fine-tuned the number most frequent words that we consider when constructing the feature vector. We tried to use top 1000, 2000, 3000, 4000 words respectively, and the result is as follows:

Table 4: Fine Tuning K

K	Validation Accuracy
1000	0.8055348490495714
2000	0.8097279165113679
3000	0.8122437569884458
4000	0.8159018143009605

It makes sense that as K increases, the validation accuracy also increases. This is because as we use longer vector to represent each text review, or capture more possible words in the review, the model could do a better job incorporating more elements and make more precise predictions.

After fine-tuning, we found the hyperparameters that give up the best validation accuracy for Unigram and Bigrams ensemble model are

- $\lambda \approx 12190$.
- $k = 4000$.

Anything above $k = 4000$ caused the kernel to crash, and thus the results of a larger k are unknown. Because of the high lambda, we expect a lot of coefficients to be closer to 0, which may not necessarily be as bad in a larger degree model since we are dealing with hundreds of thousands of words. The best accuracy we achieved was approximately 0.8159, which is a substantial improvement upon our baseline, and thus in conclusion, despite a lot of the uncertainties of the current model, can be considered successful. However, we could've used a different approach to label the text data – as rating would not be the one and the only best indicator of text review sentiment. By labeling text reviews better through more advanced approaches, we might be able to achieve a higher model accuracy.

6 ACKNOWLEDGMENTS

Thank you to Julian McAuley for a very fun Fall Quarter. We learned a lot. A special thanks to the peer readers as well.

REFERENCES

[1] Xing Fang and Justin Zhan. Sentiment analysis using product review data. *Journal of Big Data*, 2(1):1–14, 2015.

[2] Ruining He, Wang-Cheng Kang, and Julian McAuley. Translation-based recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, RecSys '17. ACM, August 2017.

[3] Jiacheng Li, Jingbo Shang, and Julian McAuley. Uctopic: Unsupervised contrastive learning for phrase representations and topic mining, 2022.

[4] Rajiv Pasricha and Julian McAuley. Translation-based factorization machines for sequential recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, page 63–71, New York, NY, USA, 2018. Association for Computing Machinery.

[5] An Yan, Zhankui He, Jiacheng Li, Tianyang Zhang, and Julian McAuley. Personalized showcases: Generating multi-modal explanations for recommendations, 2023.