Personal User Recommendation System in Yelp

Zane Alpher
za3df@virginia.edu

Muxuan Yan
my2pk@virginia.edu

David Yoon
djy6cg@virginia.edu

Abstract

Yelp is a platform that helps millions of users see which businesses are available in a specified area. Users are allowed to check and give reviews of a particular business, as well as offer tips and make reservations. Today, Yelp has over 178 million unique monthly users and is the 44^{th} most visited website in the US according to Alexa. Currently, Yelp's system makes a lot of recommendations for users based on good reviews, time, and location. However, Yelp's current recommendation system lacks personalization, and does not take into account a specific user's interests. Given these problems, in this paper, we mainly introduce building recommendation prototypes to look at a user's review and try to offer automatic personalized recommendations based on the user's interests and patterns.

1 Introduction

For Yelp specifically, the most prominent feature on the home page is the search bar. Users can search for nearby businesses, and keywords in the search query are matched against the business description and user reviews in order to find relevant results. Content found further down in the homepage is based on the user's detected location: showing suggestions for various business categories, such as restaurants, dentists, and locksmiths, as well as recent activity nearby.

In many aspects of our lives, we often interact with recommendation systems. On the internet, we find ourselves using recommendation systems to find goods that we are looking for. Our main goal in this paper is to present recommendation systems based on a user's past reviews, the star ratings that they have given, and the type of restaurants that they have visited. It may be likely that users are most interested in some type of business, e.g. a restaurant, over other types. It also seems likely to us that users would have preferences within that business category. User submitted content such as reviews and tips give an opportunity for users to express their preferences through the businesses they choose to visit. Through the analysis of user reviews, we are able to connect similar users with each other as well as identify businesses that have similar reviews. From this idea, we want to be able to recommend restaurants to users given found similarities.

2 Dataset

Yelp currently has an open dataset that can be used for learning. It is a subset of their actual dataset and contains json files of 8 million reviews over 2 million users and 200,000 businesses. Within these files, we can look at a specific user's data as well as specific review files for that user. Additionally, we can connect these reviews to a specific business. Using this data, we can look for active users who submit reviews, analyze the type of places that they often visit, and try to create a recommender system off of this.

There are three main datasets that were used in this project: review.json, business.json, and user.json. All of the reviews were contained within one json file. The business dataset includes information on store hours, average star rating, geographical information, etc. The user dataset includes information on when the user joined, the number of reviews he/she has written, as well as friend mappings. Each review contained many different features, but the features we wanted to look at were the $review_id$, the star rating for each review, the actual text of the review, as well as the $business_id$ and $user_id$ associated with each review, as this allowed us to connect the review with a business and a user.

In order to subset this data, we filtered through the reviews to only keep reviews, businesses, and users from Toronto, as this helped focus the data (restaurants and users over different cities didn't have many interactions with each other). Additionally, we wanted to get rid of the cold start problem so we only looked at users who gave >10 reviews (this was a number that we played around with) as we felt this gave us sufficient data to run our experiment. After subsetting and cleaning the data, we found that we had a dataset of 484,322 reviews from 107,350 users and 13,938 businesses.

3 Proposed Methods

3.1 Restaurant Similarity

For a general approach that is not specific to one user, we can give restaurant recommendations based on the similarity or distance between restaurants using the user scores as attributes. Using only the reviews in Toronto, a sparse matrix was constructed with each restaurant represented as a row and each reviewer as a column. Each feature in the matrix represents the stars that reviewer gave to that restaurant. From this matrix, we are able to calculate the euclidean distance between two restaurants based on their shared reviewers. By using the difference between the users' two reviews, the distance tells us whether users had a similar opinion about both restaurants (indicated by a low difference) or opposite opinions about the restaurants. These scores were then normalized by the number of shared reviewers to account for the different dimensionality of each comparison. The system takes a restaurant as input and returns the top 10 closest restaurants, however, it only considers restaurants with greater than $n \in \mathbb{N}$ shared reviewers to limit a low sample size affecting the rankings.

3.2 User Similarity

Another method we propose is to use the similarity between users to provide potentially good restaurant for a given user. First, we create a sparse matrix matching each user to each restaurant. Each entry denotes the rating a user gives to a restaurant. Such a rating is calculated through 2 components. The main component is the star a user gives to a restaurant in a review. This is the direct feedback we get from each reviews and is used as the base of sparse matrix. Then we have a indirect feedback that represents users' general taste preference. We evaluate a user' preference by 2 metrics, user's most visited category and user's highest rated category. A user's

most visited category is the category of restaurants that user gives the most review to. On the other hand, the highest rated category is the category of restaurants that user gives the highest stars on average. All restaurants, regardless of whether the user gives a review or not, in these two possibly same categories will both get a small increment in rating. Therefore, user U's rating to business B is calculated as:

$$R(A,B) = r(A,B) - Ave(A) + c * Fav_1(A,B) + d * Fav_2(A,B) \quad (1)$$

Where $r(A,B)$ is the star A give to B in a review, $r(A,B) = 0$ if no such review exists.

$Ave(A)$ is the average rating user A gives.

$Fav_1(A,B) = 1$ if B is in the most visited category of A, $Fav_1(A,B) = 0$ otherwise.

$Fav_2(A,B) = 1$ if B is in the highest rated category of A, $Fav_2(A,B) = 0$ otherwise.

After we obtain such a sparse matrix, we can calculate the cosine distance between users.

$$D(u_1,u_2) = 1 - \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}} \quad (2)$$

Where x_i is the rating $u1$ gives to the i th restaurant.

y_i is the rating $u2$ gives to the i th restaurant.

Based on this distance score, we select the top 5 users who have the minimal value of distance. They represent the most similar users to our given user. We then average the scores they give to all the restaurants our given user didn't review and rank them according to the result. In the end, we select the 10 restaurants with the highest average accumulated score to recommend to the user.

3.3 Sentiment Analysis with User Query

3.3.1 Rating Prediction: The first idea was to approach user sentiment as a classification problem, and try to predict a user's star rankings based on the sentiment that people had given for reviews. In order to do this, all of the review's texts were cleaned by removing all punctuation, extra whitespace, and format. Then, stop words which have no value were removed (words classified from the NLTK stopword library) from all of the reviews and Porter word stemming was performed which guaranteed that all words were stripped to their root. Once this was done, we had to extract the features from the vocabulary, which involved counting the frequency of each word in a Bag of Words representation and representing it as a sparse training matrix. In this matrix, the (i,j) entry represents the number of occurrences of the jth item in the ith review. After this, the models were tested for rating prediction accuracy and the results (further described in the results section) showed that although the reviews struggled with general star prediction, testing for positive vs. negative reviews (1 star vs. 5 star predictions) had very strong accuracy, indicating that our method is able to differentiate between a strong review vs. a poor review.

3.3.2 Recommendation based on Review Sentiment and User Input: The first part, although useful for seeing if text review sentiment is a good indicator for star prediction, doesn't really help the user in terms of pure restaurant recommendation. As a result, a slightly different approach was taken, where the goal was to set up a recommendation system that could take in user input of where they want to eat, and give a list of n number of recommendations that match the user's query. In order to do this, all of the reviews were combined together for each user and each restaurant and preprocessed in the same way as in the first step. Then, a feature extraction was done separately for businesses and users using a TFIDF Vectorizer, which we found was a bit more comprehensive than the raw BOW representation. Thus, from these two matrices, we are able to predict restaurant recommendations from a given user input. Details of example output as well as limitations are shown below.

4 Experiments

4.1 Data Preprocess

The dataset used in all of the following experiments is reviews involving businesses with categories of either "food" or "restaurant" in Toronto. In the whole dataset, Toronto has the most number of restaurants and most of its reviews are complete with no empty values. It consists of 13,938 businesses, 484,322 reviews and 107,350 users.

4.2 Restaurant Similarity

The sparse array of restaurants as rows and users as columns was too large to be help by a numpy or pandas array in python, even for just the Toronto data set, so we created a modified dictionary structure to represent the sparse array. Going through each user, the difference between each of their reviews was accumulated for each business in a dictionary. This dictionary was queried to find all the restaurants with shared reviewers to the input restaurant and each restaurants accumulated difference was then normalized. Restaurants with less than n shared reviewers were filtered out and the 10 closest restaurants were returned. More work is needed to help improve the selection of n but by inspection (for Toronto) an n of 30-50 seemed to be suitable. Any lower and the top of the rankings would be mostly restaurants with close to the minimum number of shared reviewers and any higher there would be too few restaurants to compare.

As an example, here are the recommendations for two different Chipotles (shortened to top 5) in Toronto.

A	Restaurant	Distance
1	Salad King Restaurant	0.085416
2	Pho Tien Thanh	0.101534
3	The Lakeview	0.108465
4	Sneaky Dee's	0.118580
5	Seven Lives Tacos y Mariscos	0.118666

B	Restaurant	Distance
1	La Vecchia Ristorante	0.125971
2	Khao San Road	0.152455
3	Big Fat Burrito	0.162162
4	Salad King Restaurant	0.162221
5	Duke of Kent	0.163935

The first was set to exclude restaurants with less than 50 shared reviewers because it had 224 reviews, meaning it shared reviewers with many other restaurants. The second example had to be set to only remove restaurants with less than 30 shared reviewers because it had many fewer reviews. From this we can also see the optimal n depends on the number of reviews (among other factors) and there is not a constant best n.

4.3 User Similarity

Given a specific user, we start by calculating its distance to other users:

```
Target user: V4TPbscN8JsFbEFiwOVBKw
distance with other users:
user: HJECayULRM-6xh2GCCvLiA Distance: 1.005975893304072
user: Wx7cbLDqYEL3_aVZwh82Ww Distance: 1.0488857627873993
user: F-XkGoU9tZxZLdGI6JAtZw Distance: 0.9985426028474751
user: V4TPbscN8JsFbEFiwOVBKw Distance: 0.0
user: PYxu2hkaolJpSru9pYLwnw Distance: 0.9699704109934997
user: yT_QCcnq-QGipWWuzIpvtw Distance: 0.9803817973546876
user: 8HCEPF0IflyB690Q745fKg Distance: 0.9591179782988739
user: _t3BJzyGaqr9mcDazYiYAQ Distance: 1.0017831491721163
user: Q9mA60HnY87C1TW5kjAZ6Q Distance: 0.9961175033591767
user: JDEhGFIPAyIMkfHD0G13UA Distance: 0.9855373085776403
user: BGzavA_ddMr-jGmhArv7fg Distance: 0.9565778456061825
```

Figure 1: the distance to other users for an example user

As we can see, the distance of a user to itself is zero and its distance to other users varies.

Then we pick 5 users with the least distance to the given user and rank the other restaurants based on the ratings of these 5 users.

```
For user:  GbDDI-KKO79yBGPis3lTLw
Recommended restaurants are:
                              restaurant_name  Average review
0                                      Wok Up             3.5
1                        Swaddee Thai Cuisine             4.0
2                                   Sushi Ken             3.5
3                         Hot N Juicy Crawfish             3.5
4                         Mancuso's Restaurant             4.0
5                 Hot Wok Chinese Restaurant               3.5
6  Ocean Blue Caribbean Restaurant and Bar             4.0
7                          Coconut's Fish Cafe             4.0
8                           LongHorn Steakhouse            3.5
9                        Black Angus Steakhouse             3.5
```

Figure 2: example recommendation

As we can see, these recommended restaurants are not necessarily the restaurants with the highest average ratings. Instead, they are the recommendations of users who we determined to be similar to the given user.

Unfortunately, since the data we have is too sparse, We cannot directly evaluate the correctness of this recommendation algorithm most of the time. What we attempted was to reserve 10 percent of the data as testing data and modify the algorithm a bit. Instead of producing just the rank of the restaurants, we predict the rating using the average score these top 5 similar users give. However, when we try to evaluate our algorithm in this way, we find that it only shows some results on the most active users (the users who leave the most reviews). Most of the time, we don't have a baseline "true" to compare our recommended results to.

```
-1.7599999999999998 0.0
1.0499999999999998 0.0
0.29000000000000004 0.0
0.8300000000000001 0.0
-1.7799999999999998 0.0
-0.6099999999999999 0.0
0.2299999999999998 0.0
1.06 0.0
```

Figure 3: Left: predicted score, Right: ground truth

As shown in the following figure, 0.0 means that the ground truth is not provided and we found that it is missing most of the time. We are only able to get some reasonable result from using the evaluation algorithm on the top 5 most active users. The result shows an absolute difference of 0.65 on average between ground truth and predicted score with a maximum difference of 1.2. Even though this result is promising, the credibility of this evaluation method is concerning. More rigorous evaluation methods are needed to tune the parameters better and confirm the effectiveness of this method.

4.4 Rating Prediction

For the first level of predictions, we used the entire review dataset of 484,322 reviews and split it into training set of 387,457 reviews (about 80 of the data) and a test set of 96,865 reviews and had models try to predict 1-5 star ratings. The learning results were not very strong, as the best model (Naive Bayes in this case) was only able to achieve an accuracy score of 66. As a result, the dataset was subsetted to only focus on 1 and 5 star reviews and give predictions for only those star ratings. We deemed this subset acceptable due to the fact that most people are likely to only leave a 1 or 5 star review if they particularly enjoyed a restaurant or hated it, which would make it easier for the models to identify user's likes and dislikes. This left us with 196691 reviews that we split into training (157352 reviews) and test (39339 reviews) datasets on a 80/20 split. The resulting confusion matrix statistics of the learning results are shown below.

Model	Accuracy	Precision (1 star)	Recall (1 star)
Naive Bayes	0.96	0.93	0.91
Linear SVR	0.96	0.92	0.92

Model	Accuracy	Precision (5 star)	Recall (5 star)
Naive Bayes	0.96	0.97	0.98
Linear SVR	0.96	0.95	0.95

From the table above, we can conclude that both Naive Bayes and Linear SVR are both able to classify 1 and 5 star ratings relatively well. It is worth noting that Naive Bayes tended to have a slight bias towards 1 star ratings (as noted by the precision and recall numbers being lower) and was able to detect 5 star ratings a bit better than 1 star ratings. This might have been because there are more 5 star reviews in the datasets and on average, 5 star ratings on average had a slightly less review length than 1 star reviews.

4.5 Sentiment Analysis Recommendation System

After creating the appropriate sparse matrices, we were able to use the feature weights to give recommendations for users based on their text input. For example, based on the input, "I want to have dinner with a really nice view," we would return a list of recommended restaurants and the matching star ratings based on key word matching from the matrix. Figure X below shows the sample output:

```
Favourites Dining Room
Buffets, Restaurants
4.0 9

Vista Eatery
Ice Cream & Frozen Yogurt, Restaurants, Fast Food, Food, Beer, Wine & Spirits
4.0 3

Hotel X Toronto by Library Hotel Collection
Hotels & Travel, Event Planning & Services, Restaurants, Hotels, American (Traditional), Bars, Nightlife, Resorts
4.5 31

Little Hut Cafe
Caterers, Breakfast & Brunch, Restaurants, Event Planning & Services, Cafes, Food Stands
4.0 4

Saks Food Hall by Pusateri's
Food, Grocery, Specialty Food, Bakeries
4.0 6

The Tea Emporium
Coffee & Tea, Food
4.0 6

Canoe
Restaurants, Canadian (New), American (New)
4.0 670

Boccone Deli & Pizza
Delis, Sandwiches, Pizza, Restaurants
4.0 23
```

Figure 4: List of Recommended Restaurants with Star Rating

As we can see, the restaurants listed are all places that users could eat dinner at in Toronto with a star rating of at least 4. Further investigation shows that these places all are restaurants that have a nice view. The picture below shows a picture from the top result, which shows that the restaurant has a nice view. This was also noted for all of the other restaurants in the top 3 recommended.

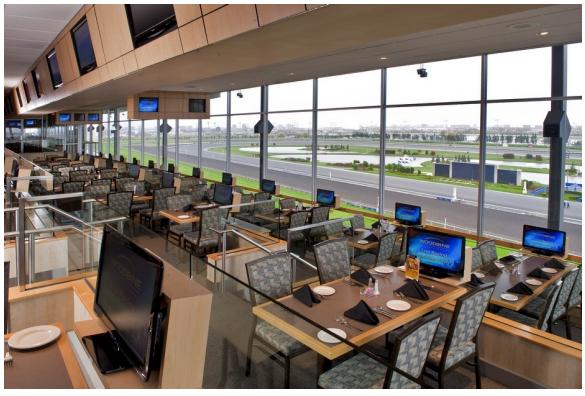

Figure 5: Top Recommended: Favourites Dining Room

Even though this algorithm seemed to work well in many cases, I noticed that the algorithm had some limitations. The main limitation was that each valid word in the user's query was not assigned a specific weight. This meant that "dinner" would have the same weight as "nice view". An example where this would fail is in a query that said "I want to have dinner at a rooftop bar. In this case, the first returned result was a Starbucks that had a coffee bar on it's top floor, which isn't what we wanted since Starbucks isn't a restaurant where a user can eat dinner. Some of the lower ranked matches accurately found what we were looking for, indicating that the weights should be distributed so that certain keywords like "dinner" or type of food have a greater influence on the search results rather than accessory traits like "rooftop".

5 Related Works

Although here are varying levels of sophistication when implementing recommender systems, but there are generally two methods that are commonly used: a content-based filtering (CBF) system, and collaborative filtering. Content-based filtering (CBF) systems generally try to model user's based on what the individual user has rated highly in the past. Generally, a content-based system requires the content of the recommended items to be largely analysable, like songs, website, images. In our cases, information directly relates to the business in the yelp dataset is rare, so it might be hard to implement in this particular case. In a collaborative setting, users with similar likes are grouped together and recommendations are given based on those similarities. Since the Yelp dataset is an open source academic purpose dataset, there was a lot of precedent work done, particularly on the review sentiment approach. There were some posts that worked on creating recommendations from user sentiment analysis, but my text processing methods differed slightly from the original posters, as well as the dataset used (previous work analyzed an Arizona subset of the Yelp dataset from a previous year). Deep learning algorithms which combine the above two approaches are also commonly used nowadays especially as the data input gets larger by the day. It gives a much finer interaction between users and items because they are non-linear, but it requires data and computing resources to train the model for it to optimize properly.

6 Conclusion and Future Work

Through our limited work, we explored several different filtering methods that were able to give restaurant recommendations to users based on past user review data. We found that if given users with a sufficient amount of review data, we could correctly identify those who share common tastes and interests and associate them together, and give restaurant recommendations based off of the restaurants that the similar users ranked highly. A similar approach was able to be taken with restaurants, where we were able to associate similarly rated and reviewed restaurants together, and give the user "similar restaurants" based off of one restaurant that was being explored. Finally, the last approach showed that text review sentiment was able to successfully distinguish a highly rated review, from a poorly rated one, and gave us a basis for creating a recommendation system that could take in a user query and return restaurants that best matched the characteristics of the query based off of previous reviews.

As for next steps, there are a lot of ways in which this project could be expanded. As discussed earlier, we believe that all 3 recommendation systems could benefit from further testing. With no confirmation evaluation metric, there's no real way to tell if a user actually believes that the recommended restaurants match what he or she is looking for.

Evaluation of recommendation algorithms is a difficult and unsolved task. There is no general formula we can apply to our recommendations to score them. In general, there are two methods for evaluating such algorithms. The first is to compare to a previous system (which we have none) to look for differences and the other is to base the evaluation off of expert rankings. The latter could work for all three recommendation systems. For the restaurant similarity algorithm users could be given a similar restaurant and fill out a survey after they eat there. A personalized version of this could be done for the user similarity algorithm as well as the text based one. To evaluate, we then compare our results to the user similarity scores using discounted cumulative gain.

Here is an example of how it would be done for the restaurant based similarity recommendation system. In this example we are requesting similar restaurants to Chipotle and having a user rate the similarity which we use as the relevance value. The system recommendations (followed by the user relevance score) are:

(1) Qdoba (4)
(2) Boylan (1)
(3) McDonalds (2)
(4) Capital Grille (0)
(5) Moe's (3)

From this, we can now evaluate and find the performance of the system. Having some user data would allow us to fine tune the system by letting us make small changes and be able to tell whether it improved the system.

$$\begin{aligned} DCG_5 &= \sum_{i=1}^{5} \frac{2^{rel_i} - 1}{log_2(1+i)} \\ &= \frac{2^4 - 1}{log_2 2} + \frac{2^1 - 1}{log_2 3} + \frac{2^2 - 1}{log_2 4} + \frac{2^0 - 1}{log_2 5} + \frac{2^3 - 1}{log_2 6} \\ &\approx 19.84 \ (DCG_{ideal} \approx 21.35) \end{aligned}$$

Additionally, our project only applied to restaurants and users in Toronto, and we believe this could easily be applied and carried out in other cities to see if results are consistent. Adding more cities my not improve the results directly because there is little similarity or crossover between cities but adding more data within a city would improve the data set. In terms of improving the algorithms itself, we believe that more features could be looked at rather than just reviews. This includes looking at the demographic data of users in order to get more accurate similarity scores between users, as well as looking into restaurant descriptions, which would help us better identify similarities between businesses. In terms of text analysis, further experimentation could be done with looking into weighing certain keywords more than others (as suggested in the results section) in order to give more accurate and focused restaurant recommendations for the user.

References

[1] Aayush Agarwal. Solving business usecases by recommender system using lightFM. Towards Data Science, June 2018.
[2] Chen Li. Prediction of Yelp Review Star Rating using Sentiment Analysis. Stanford.
[3] Theo Jeremiah. How to Build a Restaurant Recommendation System Using Latent Factor Collaborative Filtering. Towards Data Science, November 2019.