Progress Report

Yelp Restaurant Recommender System

Team

Deepak Gupta (111485745)

Jatin Garg (111462753)

Vipul Gandhi (111483224)

Answering questions posed in the Project proposal sheet :

1. Question : Are you predicting how a restaurant will be rated before seeing the reviews?
   Answer : We dropped this part of the proposal to predict ratings of a restaurant in advance and are focusing on building a recommender system for users.

2. Question : It is not clear how restaurants can target yelp reviewers since their full name is not available.
   Answer : The idea behind the posed problem was to target users for a restaurant from Yelp perspective and not from a third party recommender. As discussed during the office hours, we have dropped this problem too to focus on a concrete problem.

Keeping above points into consideration, we formulate a refined problem statement.

Problem Statement :

To build a recommender system that predicts top N restaurants for a user. The prediction is made from a set of restaurants the user has not yet visited and hence, provides the list of restaurants the user is most likely to like and give good ratings to. The system learns from past ratings given by a user to restaurants he/she has already visited.

Progress Phase 2 - Data Munging:

We use a dataset provided by Yelp for their current ongoing challenge. Though the challenge asks participants to conduct research or analysis and share their discoveries with yelp, we utilize it to build our recommender system. Yelp dataset is quite comprehensive with totaling to around 5 gbs with around **1.18 million** users, **4,700,000 reviews** by these users, details of **156,000 restaurants**. There are also over 1.2 million business attributes like business hours, parking, availability, and ambience.

Some of the important modules of Yelp Dataset:

Business.json – The dataset is about business from different categories, ranging from Automobile Industry to paint industry and health service to coaching institutes. For initial implementation we focus only on Restaurants. We manually extracted all food related businesses from the data based on the keywords(236 in total) in the category of the business we did the filtering of restaurant and finally got 75911 restaurants. All restaurants include features like, location of the restaurant, it's review count, does this restaurant deliver food or not, is outdoor seating available, noise level, good for kids, good for groups, Wifi, parking, serves alcohol, accept credit cards, waiter or self service, take reservations.

Users.json – This dataset defines all details about a user on yelp. The average stars given by user to any restaurants, number of reviews given, achievements and badges earned on yelp, number of restaurants the user is first to review for, name and location of the user as per his/her yelp profile. This data also provides friend (user id) of the user and the time since user is on yelp. There are 1183361 users in total.

Reviews.json – This data is around 3gbs and contains data in 4,700,000 rows about all reviews yelp has. The data is presented with review id, user id giving that review, business id for which review was written, the review itself, stars and reactions received on the review.

Data representation:

Restaurants are spread over 836 cities and 42 states from both United States and Canada. Distributing our restaurant dataset over cities gave very few restaurant per city. On the other hand, there were sufficiently good number of restaurants for a particular state which would have helped in better model training. So, we proceeded with restaurants in state as a metric instead of cities.
Below is representation of how we are storing our data :

- **stateToRestaurant** : A dictionary storing the list of restaurants for every state. We observed that for the given data many states have less number of restaurants and information presented by these restaurants is quite small and cannot be used for learning patterns. We set a threshold to 500 restaurants in a state and throw away rest of the states. From 42 initial states to start with, we are left with 11 states to work on. Number of restaurants in each state is listed below:
  WI - 2146, BW - 2242, NC - 5406, PA - 4977, NV - 11356, IL - 883, AZ - 16056, ON - 17318, OH - 6420, EDH - 2330, QC - 5970
- **businessToUserRating** : A dictionary storing list of tuple <userid, star> for each of the 75,911 restaurants i.e. for a restaurant, it gives us the rating a user has given to this restaurant.
- **userStateDict** : A dictionary storing all users as key and list of states the user has eaten in.

A tweak: Running Ipython notebook cells generating a big list/dict (as in our case) every while consumed a lot of time. So, we used pickle to dump all created datasets (lists,dict) into **.picklefiles** to be stored on the disk. The process is to save our python object's state in byte format so that it can be marshalled and saved on the disk. We need not run the part of code generating our dataset again and can simply load the .picklefile from disk, unmarshall it and use the returned python object. The object received is persistent in nature. We can start on from the object state we left.

Interesting observations from the dataset:

Some facts out of the Yelp dataset are stated below:

1. Out of total 1183361 users across 11 states:
   a. only 114287 users have number of reviews >= 50
   b. 214067 users have review == 1
   c. 792473 users have <= 10 reviews
2. There are 2 users who have reviews as high as 11,000.

Baseline Model:

We needed a model to compare our further models with. So, we built a very basic model (as discussed during the office hours) that would just recommend the restaurant with the best star ratings in the state for user. This model says that the user will simply give the average rating achieved by the restaurant. As it is a model including no restaurant feature or user feature, our subsequent models should beat this baseline model.

- **Evaluation Criterion:** There is no evaluation criterion for this model. Instead, we will use this model as a evaluation/comparison model for our further models.

Collaborative FIltering Model:

This recommender system model uses a collaborative filtering approach. So, we started by finding distances between users and hence exploiting users with similar features to predict restaurant ratings. For every state, we built a matrix of users versus restaurants. A cell in the matrix represents the rating a user has given to a particular restaurant where he has visited. We find similarity between users with the help of surprise library which is a Python scikit library for building and analyzing recommender systems. This would give us the ratings for all the missing cells in the matrix for a given state.

- **Evaluation Criterion:** For evaluation, we first ran the model on restaurants the user has already visited and compared ratings predicted by our model with the baseline model and the actual ratings the user gave to this restaurant. Below is a table showing the same.

User_ID	Predicted Rating By Model	Actual Rating Given by User	Average Restaurant Rating
rOhyDyMWN7ckYDNA4C-JDA	3.53	3.0	4.0
9IRuYmy5YmhtNQ6ei1p-uQ	3.96	4.0	4.5
M0cI78odeq_GKqLzk8sIrw	3.56	4.0	3.0
4i0NQ2eyuQZKpXbz8TxBEg	3.99	4.0	3.5
1Qtguu0VWSPal1faEKpsAA	2.43	2.0	3.5
M0cI78odeq_GKqLzk8sIrw	3.24	4.0	3.0
9IRuYmy5YmhtNQ6ei1p-uQ	3.92	5.0	3.5

We performed the above case for a number of users spread over different states. The predicted ratings were close to the actual ratings for most of the users. Though the predicted ratings and the actual ratings differed even by a rating of 1 in some cases, still our model performed better than the baseline model.

Next, we evaluated predicted error values by this model with the values predicted by our baseline model for every state. We also choose this as our evaluation criterion for our later advanced models. Below is a table showing results we got.

State	No. of Users	Mean Absolute Error (Collaborative Filtering Model)	Mean Absolute Error (Baseline Model)
QC	119363	0.60495	0.81798
ON	489047	0.64720	0.89923
NC	214139	0.65225	0.95367
BW	28917	0.64909	0.86445
EDH	35221	0.54461	0.70260
OH	183730	0.66958	0.95613
PA	169928	0.64581	0.92827
WI	80998	0.64376	0.90894
IL	26836	0.68731	0.97087
AZ	989170	0.63874	1.00581

Model with feature vectors:

Collaborative filtering model only takes distance(similarity) between users into account . Other restaurant features and primarily distance between user and restaurant are not captured. So, we built another model capturing these parameters and then we compared this model with both our baseline model and Collaborative filtering model.

For this model, we determine that for a user to visit a particular restaurant, two important aspects to focus on are the features of the restaurant (ethnicity of cuisines, desserts available, drinks available) and the location of the restaurant (i.e. nearby to the user location).

We will first talk about the features of the restaurant that we have taken into consideration and then we talk in detail about location.

We listed all the given properties of a restaurant and called it restaurant feature vector. For every restaurant vector, there were 130 columns to start with. For all the features a particular restaurant has, we put a 1 in the column value while other columns are 0. In order to reduce our feature dimensions (currently 130), we performed manual feature selection. We observed that some of the features from dataset had very low counts and can be dropped while many features of importance with low counts can be merged together to form 1 feature with high value count.
We filtered categories on the basis of ethnicity of cuisines, whether it is a restaurant, bar, cafe or dessert place. While filtering categories on the basis of ethnicity of cuisines, we distinguished cuisines on the

basis of country of their origin and popularity. Example : Falafel is categorised as Middle Eastern while Soba and Sushi are categorised under Japanese food. We've also built the user vector(talked in detail below) with the same set of features (same dimension vector) as that of a restaurant. When a user visits a restaurant the features of that restaurant gets added to the user vector.
So the user feature vector particularly shows the type of food, cuisine and restaurant features he/she likes. As the number of restaurants offering cuisines of a particular country was less in count, we figured out that merging cuisines of nearby countries could both solve our problem and is logical as such foods are much similar in taste. Arab food, Iranian food and Israeli food can be safely categorised into Middle East food while Indian,Pakistani and Nepali dishes can be categorised into Indian Subcontinent food. So, a restaurant offering either Indian or Nepali cuisine will go to the Indian Subcontinent category.
Below are some of the merged cuisines on the basis of countries :
south_america = ['cuba','haiti','venezuela','colombia','caribbean','puertorico','hawai','peru','latin']
indian_sub = ['india','pakistan','nepal']
china_nearby = ['taiwan','china','korea','japan']
rest_asia = ['asian','cambodia','thailand','malaysia','laos','vietnam']
rest_europe = ['ireland','portugal','czech','germany','belgium','greece','spain','scandinavian', 'ukraine','hungary','france','england','switzerland','poland']
combine_canada_america = ['canada','america']

Similarly, we merge the features where we were given details about type of parking available for the restaurant (garage,lot,street,valet) and formed 1 combined feature named 'has_parking'. While individual categories of music played in the restaurants like karaoke, dj, jukebox, live, video were merged to a category named 'has_music'.
Also, we dropped a number of features on the basis of threshold. Thus, we reduced our initial 130 columns to 49 columns in total to be used in our feature vectors.

Another important aspect determining a user visiting a particular restaurant is that the restaurant should be in a nearby area to the user. Thus, we have also taken location into consideration. To find nearby restaurants for a user, we set a distance of 10 miles from a user to pick restaurants from. For this, we need both restaurant location and user location to find restaurants in the 10 mile range. Though we had available information for location of every restaurant in the dataset but we did not have location for users.
For calculating the location of user, the only available parameter was the location of restaurants the user has eaten at. We formed clusters for these restaurants on the basis of how distant/close are 2 restaurants. If 2 restaurants are less than 10 miles apart, we merge them, give them a new name and add the number of times user has visited each of these 2 restaurants. We continue doing so for a user till no further merging can be done. Now we have map of all nearby merged restaurants along with number of visits by this user for all restaurants in this merged set. On the merged restaurants, we find the merged_restaurant with maximum user visits. We say the location of this merged_restaurant to be the user's location as the user is eating in nearby restaurants, the user is more likely to be situated nearby(maybe work or home). On further analysis of our dictionary storing all restaurants for a given user, we found out that there are some users (58,739) who have eaten at restaurants in quite far off locations(multiple states). Possible reasons for this can include often holiday or business trips, different

workplace state(weekdays) and home state(weekends). For such users, we gave the user a location for every state he has eaten in and similar process as above is performed for calculating user's location.

After this, we are ready with our Restaurant vector. Now, we build feature vector for every user. As there are way lot more users, the user vector is calculated on the fly when needed to make predictions for that user. A separate model is trained for every user in the state. The dimension of features for the user vector are equivalent to that of restaurant vector. For a given user, we pick every restaurant the user has visited. For every such visited restaurant we encounter, we add the feature vector of that restaurant to our user vector.

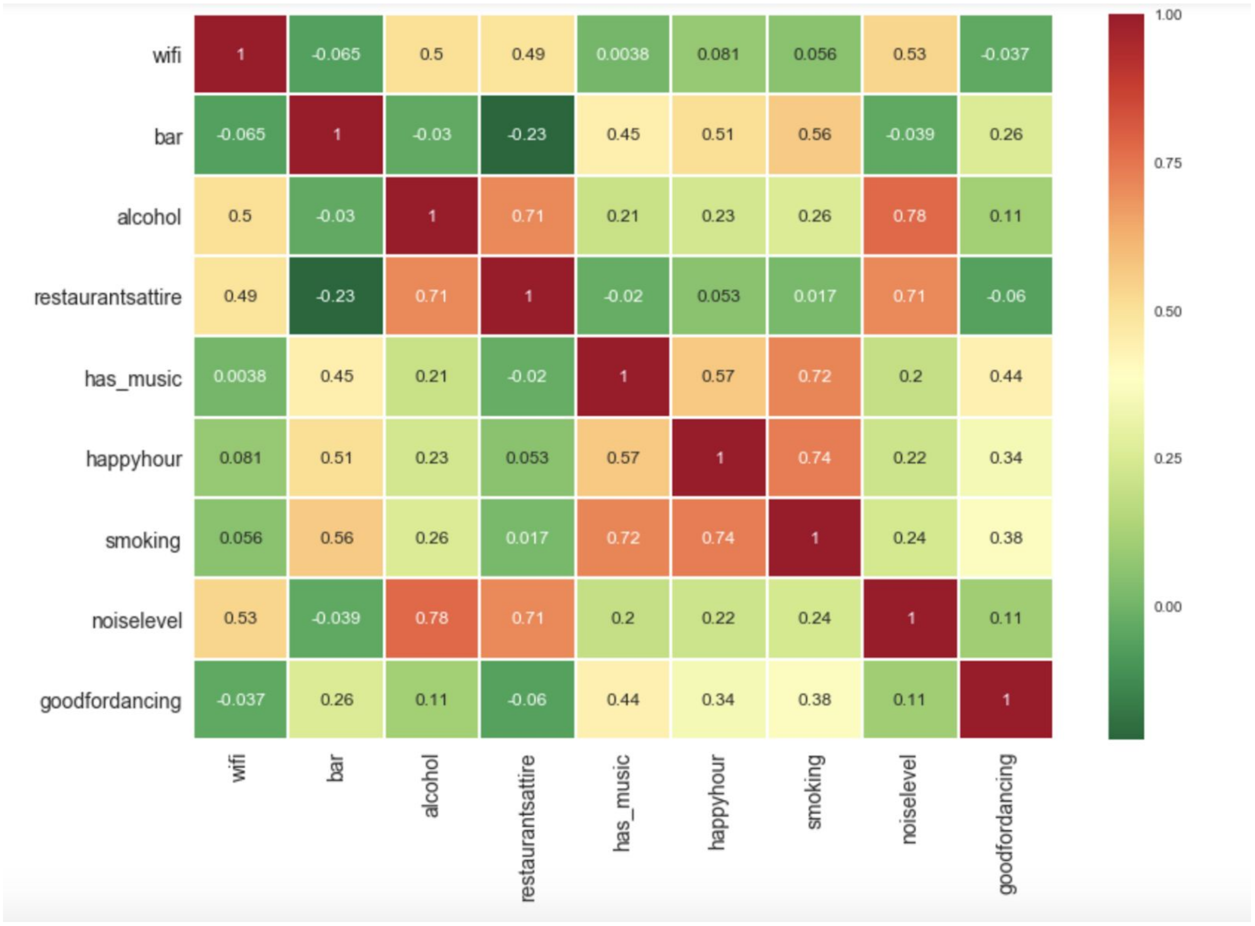
$$U < f1, f2...., fn > \; = \sum_{i=1}^{\#\, res\ visited\ by\ U} R_i < f1, f2, ..., fn >$$

Thus, the user vector will grow every time a new restaurant is visited by the user. Now, for every user we find all restaurants and generate a score for them using the below function.

```
def getScoreForRestaurantByUser(userVector,resVector,userID,resID,state):
   alpha = 0.5
   beta = 0.5
   score = (alpha * getCosineSimilarity(userVector,resVector)) + (beta *
distParameter(userID,resID,state))
   return score

def distParameter(userID,resID,state):
   userLocation = getUserLocation(userID,state)
   resLocation = resLocationDict.get(resID)
   dist = getDistance(userLocation[0],userLocation[1],resLocation[0],resLocation[1])
   R = 6373.0
return (1-(dist/R))
```

- **Interesting observations:** Some of the interesting observations for the feature vector model, based on the correlation heat map plotted for our reduced dimensions are as below:
    a. **Correlation between 'Alcohol' and 'Noise Level' = 0.78**
    Our dataset shows that places where alcohol is served are typically more noisier than others. As per surveys, people tend to avoid restaurants with high noise levels.
    b. **'Smoking' and 'hasMusic' = 0.72**
    People like to smoke with some music and Smoking places/joints normally have good music. Results even show that preferences for classical music are associated with lower smoking, while preferences for bluegrass, jazz, and heavy metal music are associated with higher smoking.[Ref : https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3160811/]



- **Evaluation Criterion:** For now, we have not recommended top restaurants using this model (to be done in further phases). We have just tried to validate the model first.
  A problem here is that we only have the scores but not the actual rating a user is expected to give to a restaurant. For evaluation, we need to convert our model generated scores to ratings. For this, we first run our model through all the restaurants the user has visited. For these restaurants, we now have the actual ratings given by the user and scores predicted by our model. Training these values through a Random Forest Regressor, a correlation is formed and we use this regressor model to convert our predicted scores to predicted ratings for unvisited restaurants.
  Evaluation criterion for this model remains same to the evaluation criterion adopted for collaborative filtering model ,i.e. for every state, we compare mean error from both the current model and the baseline model.
  For building learning patterns from user vectors, it is important to choose a threshold for the restaurants a user has rated. We have only taken those users which have visited a minimum of 10 restaurants in the state. Below is a table for Mean Absolute errors.

State	No. of Users (visited_res >10)	Mean Absolute Error (Feature Vector Model)	Mean Absolute Error (Baseline Model)
NC	3505	0.52216	0.89795
NV	6957	0.52002	0.89111
WI	1391	0.50203	0.83280
BW	496	0.50925	0.81861
EDH	522	0.42373	0.66215
IL	465	0.52577	0.88443
PA	2742	0.51154	0.86078
OH	3095	0.52726	0.89258
QC	1804	0.47232	0.75635

Future Work:

- For cases with cold start where a user has visited very few restaurants, we plan to use the Yelp friend data which is available for each user.
- For states with less restaurant, we will work on finding external data (scraping from tripadvisor, zomato, facebook) to merge with our yelp dataset so as to increase the number of states we are currently working on.
- Building a advanced model.