Yelp Recommendation: A Comparison of Clustering and User-to-user Algorithm

Final Report for CMSC 12300
Ellen Hsieh, Fulin Guo, Jiaxu Han, Ying Sun
Github Repository: https://github.com/sunying2018/CMSC12300_Project

Dataset

We used the "Yelp Open Dataset" for this project. The size of the original dataset is about 3.6GB when compressed and 8.69GB when uncompressed. There are six JSON files in total. Each file is composed of a single object type, one JSON-object per-line.

Yelp Open Dataset includes 6685900 reviews, 192609 businesses, 200000 pictures, and 10 metropolitan areas. It also covers the 1223094 tips by 1637138 users, over 1.2 million business attributes like hours, parking, availability, and ambiance aggregated check-ins over time for each of the 192609 businesses.

The dataset we mainly used is the review file (review.json) from the Yelp Open Dataset. The review file contains about 6.8 million entries of reviews. In addition to the actual review text, it also contains the user ("user_id") who wrote the review, the business ("business_id") that the review is written for, and the rating score ("stars", ranges from 1 to 5) that reviewer gave to the business. It also includes the date of the review and various kinds of votes received. We used jq, a command line json processor, to convert the review.json into a .csv file and we extracted only three variables — "user_id", "business_id", and "star" — for further analysis.

The rating score distribution for the whole dataset (6,685,900) is shown in Figure 1.

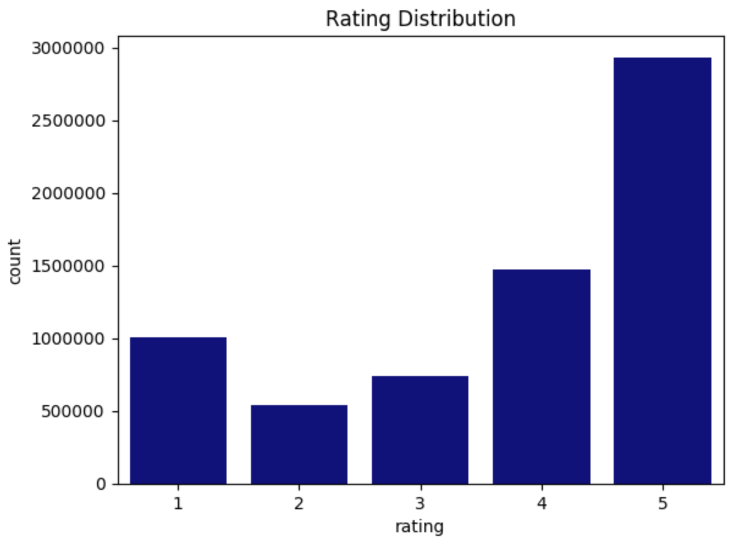

Figure 1. Whole Dataset Rating Score Distribution

Hypotheses

1. Some people have similar taste in choosing restaurants.
2. The user-to-user algorithm has a higher prediction accuracy than the clustering algorithm for recommending the business for users.

It is assumed that two or more people have similar tastes if they both visited some restaurants or businesses and gave similar ratings to them. In addition, assuming that two people, such as A and B, have similar tastes, it is more likely that A would also like to visit businesses that B has visited, and vice versa. Based on this hypothesis, we could recommend businesses to a user according to the businesses visited by similar users.

We implemented two algorithms based on the above assumption: the clustering algorithm and user-to-user algorithm. In the user-to-user algorithm, it compares the similarity of every two users in the dataset. The clustering algorithm, according to its name, classifies users into clusters and seems to be a rougher estimation than the user-to-user algorithm. Therefore, we hypothesized that the user-to-user algorithm has a higher prediction accuracy than the clustering algorithm.

Algorithm

User-to-user Algorithm

The user-to-user algorithm uses a comprehensive similarity score to tell whether a two-user pair is a similar pair. Group by the *business_id*, we iterate over all possible user pairs where both users have been to the same business and we calculate the similarity score for every two users. The similarity score is the weighted sum of the frequency difference and rating difference between the two users. The similarity score is defined as follows:

$$\text{freq}\_sim = \frac{\min\{visits1, visits2\}}{\max\{visits1, visits2\}}$$

$$\text{rating}\_diff = \frac{|rating1 - rating2|}{5}$$

$$\text{similarity}\_score = \text{freq}\_sim \times w + (w - 1) \times \text{rating}\_diff$$

The above equations are used calculate the similarity score between two users. Visits1 and visits 2 represent the visiting times for the business, rating1 and rating2 represent their average rating for the business and the w in the similarity_score represents the weight for the freq_sim.

The optimal parameter we selected here is the weight of the frequency. After calculating the similarity scores for all pairs of users, we selected either 3 users or 5 users that have the highest similarity scores with respect to each user.

We split the whole dataset into the training and testing dataset (70% in the training dataset and 30% in the test dataset), calculated the prediction accuracy and selected the optimal parameter. More specifically, to evaluate our model's performance and test the hypothesis 1 (whether some people have similar taste in choosing restaurants), we also calculated the baseline accuracy without using the current user-to-user algorithm. That being said, we assumed that everyone is similar to each other, no matter how many times they have visited the same businesses or how many stars they have given to them. For a given person, we recommended all businesses visited by anyone else but himself/herself to him/her. In such a way, we were able to know whether the prediction accuracy of the clustering model has improved or not.

At last, we used a similar algorithm to find the most similar user for each user. We wanted to recommend similar users to each other in order to provide a potential social opportunity in real life.

Clustering Algorithm

The clustering algorithm aims to classify users and to make recommendations for each user based on other users who are classified into the same cluster. First, for every business, users who visited it more than certain times and rated this restaurant higher than a certain rating score are grouped into a sub-cluster. The sub-cluster is not the final cluster, but an intermediate set that is used to create the final classification. Then, users who are put into the same sub-cluster for more than certain times will be put into the same final cluster. Finally, we recommended the businesses that were visited by all the other users in the same final cluster to each user.

We used a test set to check whether users in the same final cluster are actually similar to each other. The criteria to check user similarity is the same as the criteria used in the user-to-user algorithm to calculate the prediction accuracy in the test set. We also calculated the baseline accuracy without using the current clustering model in the same way as the user-to-user algorithm.

Prediction Accuracy

There are multiple potential ways to calculate prediction accuracy. Here, we defined that a prediction is correct (i.e. two users are actually similar to each other) if *freq_sim* is greater than or equal to 0.5 and *rating_diff* is less than or equal to 0.2 between the two users. Otherwise, we deemed that the prediction is wrong. The *freq_sim* evaluates the similarity between two users based on the frequency of visiting the businesses and the *rating_diff* is the difference between two users with their ratings on the same businesses that they went to.

Big Data Approach - MapReduce :

User-to-user Algorithm:
We used a four-step MapReduce to implement the clustering algorithm:

1st step: We computed all users for each business and recorded their ratings and visiting times for each business. Then, we constructed a data structure to record all the information that we need to track.

For instance, the key here is a *business_id* and the value is a list like this: ['FL2j0euK58x-JDwdolhIKg': (4.0, 1), 'U0WU9d6gpMNkO_tPXK58Sg': (3.0, 1)]. This list is a record for one business in our code, it includes all the *user_id*s and a tuple to indicate the average rating and visiting times for this user to this business.

2nd step: We gave different weights for ratings and visiting times to construct similarity scores for each user pair. Then, we used a random number in this step to determine whether this business should be assigned to the training set or the testing set. Finally, we yielded the user pair as key and the similarity score as the value. If the business is assigned to the training dataset, we calculate the similarity score of the pairs who visited this business, the range of the similarity score is higher than -1. On the other hand, if the business is assigned to the testing dataset, we assign the similarity score -2 if the pairs are truly similar and assign it -3 if they are not truly similar. Since the ranges of similarity scores when calculated for the training set and for the test set are not overlapped, we could tell whether the value is from the training set or from the test set.

3rd step: Find the Top K most similar users based on similarity scores.

4th step: Compute the accuracy scores to evaluate our model's performance compared with the prediction accuracy calculated by considering all the user pairs who came to the same business as similar users. Then, we select the parameter to determine the best weights split between ratings and visiting times to gain the highest prediction accuracy.

Clustering Algorithm:
We use a three-step MapReduce to implement the clustering algorithm:

1st step: The same as the first step of the user-to-user algorithm.

2nd step: we first use the random variable generated by python to decide whether the business should be put in the training set or the test set. If the *business_id* is supposed to be put in the training set, we decide which users who visited this business should be classified in the same sub-cluster. If the business set is in the test set, we calculate whether the two users are truly similar in selecting this restaurant. This step yields the list of sub-classifications corresponded to each the business in the training set, and the number of successful predictions for each restaurant if the business is in the test set.

3rd step: For the test set, we first calculate the accuracy rate without utilizing our model. Then, we used the training set to combine sub-classifications to generate the final classification of users based on the times threshold. Then, we use the test set to calculate the accuracy rate of our model and compared it to the accuracy rate without using the model. which is further used to select the best parameters.

Runtimes：

We used Google Cloud to run the codes. It took 3-4 hours for each user-to-user algorithm and about 30 minutes for each clustering algorithm using 16 CPU, 6 cores, 50 GB disk instance.

Results:

Table 1. The Comparison of User-to-user Models and Clustering Models

Algorithm	User-to-User	User-to-User	User-to-User	User-to-User
Parameter	Frequency weight: 0.5 Rating weight: 0.5 Top 5 similar users	Frequency weight: 0.7 Rating weight: 0.3 Top 5 similar users	Frequency weight: 0.7 Rating weight: 0.3 Top 3 similar users	Frequency weight: 0.9 Rating weight: 0.1 Top 3 similar users
Prediction Accuracy	Prior: 0.667 Post: 0.664	Prior: 0.667 Post: 0.674	Prior: 0.667 Post: 0.677	Prior: 0.667 Post: 0.680
Algorithm	Clustering	Clustering	Clustering	Clustering
Parameter	Frequency threshold: 1 Rating threshold: 2 Times threshold: 1	Frequency threshold: 1 Rating threshold: 3 Times threshold: 1	Frequency threshold: 1 Rating threshold: 3 Times threshold: 2	Frequency threshold: 1 Rating threshold: 4 Times threshold: 1
Prediction Accuracy	Prior: 0.669 Post: 0.761	Prior: 0.673 Post: 0.750	Prior: 0.689 Post: 1.000	Prior: 0.674 Post: 0.777

1. We could see from table 1 that, in most instances, the prediction accuracy improved after using the recommendation algorithm.

2. The clustering algorithm can increase the prediction accuracy by about 10%, which is much higher than the user-to-user algorithm.
3. The parameters that result in the highest prediction accuracy increase for the user-to-user algorithm is *Frequency weight=0.9*, *rating weight=0.1*, and use the top 3 similar users.
4. The prediction accuracy increase for the user-to-user algorithm is higher when we consider the 3 most similar users than when considering 5 most similar users, which is reasonable.
5. The parameters that result in the highest prediction accuracy increase for the clustering algorithm is *Frequency threshold=1*, *Rating threshold=3*, and *Times threshold=2.* This makes sense since when *Times threshold=2,* we put users be in the same final classification only if they exist in the same sub-classification larger than 2 times, not 1 time.
6. The results show that people might have similar tastes and different tastes with others in choosing restaurants, in other words, people are not totally similar or totally different to each other within the group for the taste of choosing restaurants. This is because at least one of our algorithms significantly increased the prediction accuracy.
7. The clustering algorithm performs better than the user-to-user algorithm.

Conclusions

The purpose of our project is to provide recommendations for yelp users. For each user, we recommended them some businesses that they have not visited but the people that are similar to them have. One hypothesis is that there exist some similarities in different users. Some groups of users might like certain kinds of restaurants, while the other groups of users might like going to other kinds of restaurants.

We used two algorithms, user-to-user and clustering algorithm, to prove our hypothesis. For the user-to-user algorithm, we calculated similarity scores for every pair of users and then for each user, then we ranked his/her similarity score with respect to all the other users and recommended the 3 or 5 most similar users to him/her. For the clustering algorithm, we first classified users into different clusters. Then, for each user, we recommended businesses visited by all the other users that are in the same cluster with them. These two algorithms have different computation complexity. Since we need to compute the similarity score for all pairs of users, the computational complexity of the user-to-user algorithm is larger than that of the clustering algorithm. On the other hand, the clustering algorithm does not compare all pairs of the user, it only classifies users so we have the hypothesis that the user-to-user algorithm can perform more accurate prediction than the clustering algorithm.

We used mapreduce to implement the above algorithms and used Google Cloud to run the codes, it took 3-4 hours for each user-to-user model and about 30 minutes for each clustering model using 16 CPU, 6 cores, 50 GB disk instance. The results showed that people might have similar tastes and different tastes with others in choosing restaurants, in other words, people are not totally similar or totally different to each other within the group for the taste of choosing restaurants. This is because at least one of our algorithms significantly increased the prediction accuracy. Another finding is that the clustering algorithm performs better than the user-to-user algorithm. Especially, when we set the times threshold to be two, the prediction accuracy became 100% using our prediction accuracy calculation criteria.

For future work, we might find ways to iterate MapReduce to conduct the k-means clustering method to classify users. The iterative MapReduce could also be used to find optimal parameters from a larger range of alternative parameters. In addition, we might also take users' review into consideration while making recommendations. To be more specific, we might implement some text analysis such as TF-IDF to analyze the content of each review and based on the result of this analysis to find out the most similar users for each user, then we can make restaurant recommendation in a different way.

Challenges
In the stage of preprocessing our dataset, the first issue we met was to unzip the compressed large dataset. Since local virtual machine doesn't have enough memory to store the uncompressed file, we need to unzip our yelp data on Google cloud instance. Therefore, we created a google bucket and upload our tar zip file to it. Then, we use a shell command to unzip our large data on a Google cloud instance. The next problem we have is to convert the review.json file into CSV file. Because the file is still too large (about 5 GB), we could not open it, nor could we write a python script to convert it. Therefore, we used jq, a command line json processor, to convert the review.json file into the review.csv file.

While running our mrjob code with the whole dataset on the instance, we kept getting the permission denied error even though we already upgraded our instance with 16 CPU, 6 cores and 50 or 100 GB disk. We also submitted an application to upgrade the limit quota. However, google cloud could not answer our request promptly. Therefore, we only used a subset of the original review.json dataset, 1 million data, to implement the code. Although we only run the subset, the run time is still quite long. For the user-to-user model, it took about 2-3 hours to finish one model. For cluster model, we can only use 100,000 review data also due to limited quota and it took about 30 minutes for each cluster model.

We spent a significant amount of time trying to define "similar users" and there seems to have many ways to do it. We figured that the best way to get the optimal parameters would be to split the dataset into training and testing dataset and do a cross-validation. However, given the run time of each model, we can only test 8 models with different set of parameters. Again, if we were able to get permission to up the limit quota, we might be able to test more models. Apart from the run time, it was also challenging for us to find the best way to split the dataset into training and testing dataset. The challenge is that we could not find a practical way, neither using python script nor using shell command, to split the *business_id* randomly while put other variables (user_id and *star*) into two CSV file. Eventually, we used random number and did a train-test split in one step of the MapReduce.

As for the challenges for our algorithm, at first, we try to recommend businesses to users using mrjob. However, our initial intuition was to combine two results from two MapReduce results, the most similar user pairs and the restaurants recommended by users. However, we could not figure out a proper way to merge two large data file. We tried to use a shell command, *join*, to join two output CSV files using user ID as keys. The result of using "join" seems to work but the format is not right. Instead of properly insert the output data into the right column, everything is inserted into one column. Therefore, we could not successfully merge two data file and then make the restaurant recommendation.

Another challenge we met was implementing the k-means method using MapReduce. What we designed is to update the k means vector in each MapReduce. That is, each MapReduce yields the updated k means vectors, and then these vectors are passed onto the input of the next MapReduce. However, we did not find out a way to iterate our mrjob to implement k-means in this way.

References

Yelp Open Dataset. (n.d.). Retrieved from https://www.yelp.com/dataset

Introduction to Recommendations with Map-Reduce and mrjob. (n.d.). Retrieved from http://aimotion.blogspot.com/2012/08/introduction-to-recommendations-with.html

Bodoia, Max. "Map-Reduce Algorithms for k-means Clustering." accessed online at: https://stanford. edu/~ rezab/classes/cme323 S 16