Dining Recommendation System

Group Members: Nora Francis, Hiya Chirag Thakkar,
Julia Sami Moheb Helmy Youssef Zaki, Shannon Arlene Tanoto

March 2024

1 Introduction

Toronto is a city full of numerous restaurants of varying cuisines. Consequently, it is often difficult and time-consuming to pick out a restaurant that fits our preferences. Our program facilitates the process of choosing a restaurant based on the user's preferences. We want to apply what we've learned in CSC110/CSC111 to this real-world problem.

This raises a valid question, what makes our program different from using Google or other search engines to find highly reviewed restaurants compatible with the user's preferences? This program uses a collaborative ranking system. First, the system works by matching the user to other users (stored in the database) who share similar dining habits and ranks these users in terms of their preferences' similarity. Next, this order of users enables the program to return a list of highly-reviewed restaurants visited by mutual users in such an order that it should best complement the user's preferences. In addition, this program will return several more details about each restaurant and a graph illustrating the network of user-restaurant reviews. **By implementing this system, we will be able to create a program that provides users with recommended restaurants based on their input preferences and reviews from mutual users.**

2 Datasets used

This program uses 5 different datasets found online[1] :

1. Geospatial Coordinates Database (Geospatial_Coordinates.csv): Containing all of Toronto's postal codes with their respective latitude and longitude values.
   (a) Postal Codes - Column 1
   (b) Latitude Value - Column 2

[1]UCI Machine Learning. "Restaurant Data with Consumer Ratings." Kaggle.com, 2017, www.kaggle.com/datasets/uciml/restaurant-data-with-consumer-ratings.

    (c) Longitude Value - Column 3

2. User Profile Database (userprofile_new.csv): Store users who have given reviews to restaurants they visited, together with their preferences/dining habits.
    (a) User ID - Column 1
    (b) Smoking habit - Column 2
    (c) Drinking habit - Column 3
    (d) Dress Preference - Column 4
    (e) Company Preference - Column 5
    (f) Budget Level - Column 6
3. User to Restaurant Reviews Database (rating_final_edited.csv): Compiles the given reviews/ratings of each user to a restaurant
    (a) User ID - Column 1
    (b) Restaurant ID - Column 2
    (c) Review Score - Column 3
4. Restaurant Database (restaurant_data.csv): Store details of restaurants in Toronto.
    (a) Restaurant ID - Column 1
    (b) Restaurant Name - Column 2
    (c) Restaurant's Postal Code - Column 3
5. Restaurant's Cuisine (restaurant_cuisine.csv): Compiles each restaurant's provided cuisine.
    (a) Restaurant ID -Column 1
    (b) Cuisine - Column 2

These datasets were derived from a website (UCI Machine Learning) however, they were manipulated in MS Excel to filter out data that were needed in our program.

3 Computational Overview

This program uses a collaborative ranking system that first compares the user to other users based on different dining habits, including smoking habits, drinking habits, dress preferences, company preferences, and budget level (stored in the database: User Profile Database [userprofile_new.csv]), then ranks these users in terms of similarity.

In this project, we will use a graph to store the restaurants and users who are then connected by their ratings (stored in the database: User to Restaurant Reviews Database [rating_final_edited.csv]). The vertices can either be a user class or a restaurant class, and the edges will connect a user to the restaurants they have rated. The users will be asked a list of questions about their preferences/dining habits. The user class will store the User ID and all these user preferences as its attributes. The restaurant class will store the Restaurant ID, name, cuisines they serve (stored in the database: Restaurant's Cuisine [restaurant_cuisine.csv]), and postal code (stored in the database: Restaurant Database [restaurant_data.csv]).

The similarity list between the user and a stored user in the database is calculated by several steps:

1. First, we rank each preference with these scores:
    (a) Smoking Habit: non-smoker = 0; smoker = 1
    (b) Drinking Habit: non-drinker = 0; casual drinker =1; heavy drinker =2
    (c) Dress Preference: informal = 0; smart casual = 1; formal = 2
    (d) Company Preference: solitary = 0; friends = 1; family = 2
    (e) Budget Level: low = 0; medium = 1; high =2
2. We find the absolute difference in each preference between our program user and a user in the database, and store each value in the similarity list. For example: for user1 with preferences [0,0,2,1,1] and user2 with preferences [0,1,1,1,1] the similarity list will be [0,1,1,0,0].

Once we get the similarity lists between our program user and each user stored in the database, we sort the lists in descending order according to the number of zeros (which refers to the number of matches in preference). If two lists have the same number of zeros we break the tie by comparing the sums of the lists, the list with the lower sum comes first. This ranks the database users according to similarity with the program user. This allows the program to go through the database users in order of similarity and look for their vertex neighbors (restaurants) with edges that have 4 or 5 star ratings. Then, return a number of recommended restaurants based on a limit that the user chooses. So the recommended restaurants you get back were only given either 4 or 5 star reviews by your mutual/similar user. Note that this may be different from the average ratings (discussed below). We found that having someone who has the same preferences as you give these restaurants 4 or 5 stars most likely means you will also give them 4 or 5 stars, regardless of what the average rating is.

Furthermore, with the graph, we can calculate the average ratings of each restaurant returned by averaging the edges' weight of the restaurant vertex's neighbors. Besides that, we also ask the user to input their postal code which is then used to calculate the distance between the user and each of their recommended restaurants. To calculate this distance we first collected the latitude and longitude values (stored in the database: Geospatial Coordinates Database [Geospatial_Coordinates.csv]) of the user's postal code and the restaurant's postal code. Note that the code only checks the first three letters of the inputted program user's postal code. So in the case of the postal code we proposed you input 'M5S2E8', only 'M5S' will be looked for in the Geospatial_Coordinates.csv file. Once it is found, the code makes use of the longitude and latitude associated with it (in that same row of the file) in the distance calculation. The same process happens but with each recommended restaurant whose postal codes are amongst their attributes. Then, the distance between the two locations is calculated with the following formula:

ACOS((SIN(RADIANS(Lat1)) * SIN(RADIANS(Lat2))) + (COS(RADIANS(Lat1)) * COS(RADIANS(Lat2))) * (COS(RADIANS(Lon2) - RADIANS(Lon1)))) * 6371

Notes: acos: cosine inverse; lat1: user's latitude; lat2: restaurant's latitude; lon1: user's longitude; lon2: restaurant's longitude; 6371 is the Earth radius in km.

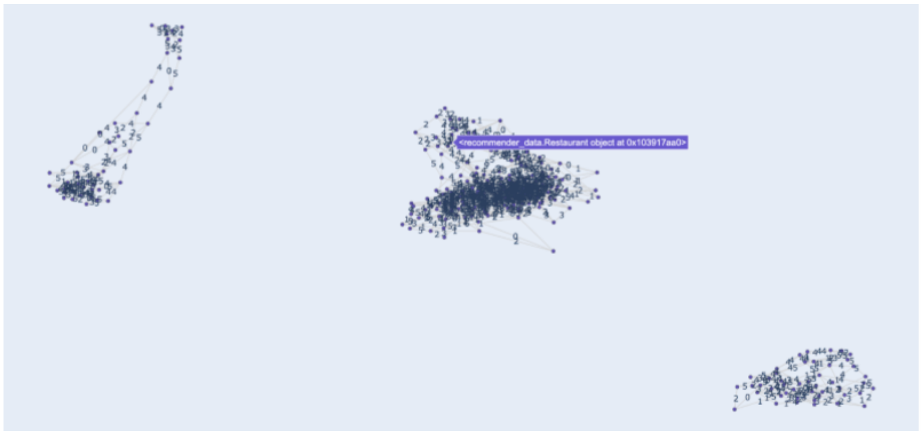
This program uses several Python libraries such as:

1. Networkx: This library is used for the visualization for the graph itself in the visualization.py file.
2. Plotly: This library was used to display the results of the recommended restaurants that the user gets from the program. We used bar charts that plots the data based on the average restaurant rating and their distance from the user to display this data and the code for this can be seen in main.py.
3. Pandas: This library was needed to run plotly
4. Math: This library is used in the get_distance function in the computation_compiling file where we use the formula mentioned above to find distance between a given restaurant and the user.
5. csv: this library is used to read the csv files into python so we load the data into graph and the classes.

Combining all these computations, the program provides a ranked list of restaurants based on the program user's preferences, alongside the average rating and also how far the restaurant is from the user's location. Besides that, a graph as a visualization of the networks of user-restaurant reviews will be outputted.

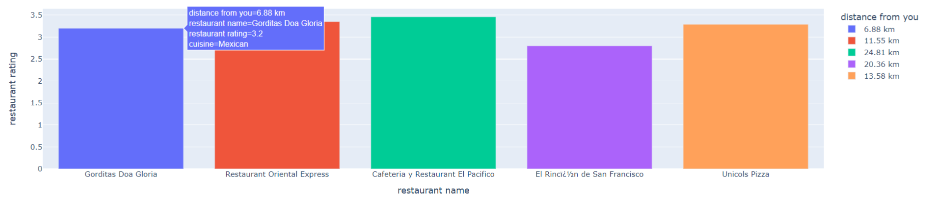
4 Instructions for obtaining datasets and running the program

Install the Python libraries listed under the requirements.txt file. The instructions for running the program are shown under the program_run.txt file. You must first, however, download the files from the Data Files zip file uploaded to MarkUs which contains all the data sets mentioned in part 3. You should uncomment lines 21 and 22 of the main.py file to visualize the graph, one of our visualizations. Once you run main.py, this graph will appear. It shows all the users, the restaurants they reviewed, and the ratings they gave each restaurant. Note that this may take some time. Here is what the graph should look like:



```
<recommender_data.Restaurant object at 0x103917aa0>
```

Visualizing this graph would have taken you to window in your browser so once you go back to Pycharm, you will be prompted with a series of questions. This where the fun starts! As said, the specific answers you should give are all listed in the program_run.txt file submitted in MarkUs. After you answer the last question, a bar graph - as the one in the screenshot below - will appear in a window in your browser. It shows the recommended restaurants and their average rating (this average rating is calculated from the ratings that each user who reviewed this restaurant gave). When you hover over each bar, you see, as shown in the screenshot below, the distance between this restaurant and your location - which you imputed when you were asked for your postal code. You will also see the restaurant's name, average rating and cuisine. Seeing the cuisine is an extra feature we added to allow the user to know what cuisine his restaurant serves and thus improve their resturant selection process. Expect to get back the exact restaurants you see in the screenshot below - restaurant names are also listed in the program_run.txt file. You can of course try inputting different answers and you will get recommended restaurants every time!



5 Changes in the project plan

Initially we wanted to use a decision tree to find users with preferences that are an exact match to the program user, and recommend restaurants that these users rated highly. After reviewing our data set and based on the feedback we received from our TA, we decided to use a similarity based approach to finding similar users because our data set is small and an exact match can't always be found. Another change we made was to use plotly instead of tkinter to display the data. Our initial plan was to use tkinter to create a simple graphical user interface that shows the recommended restaurants. However, based on the feedback we received, tkinter was said to be slightly harder to implement and understand and we wanted to focus more on the implementation of the data structures and finding similar restaurants and users. Therefore we switched to plotly instead. We also did not filter the recommended restaurants based on their distances and instead simply returned the distance between the user and restaurant when displaying the data. This was due to the limited dataset and filtering through the recommended restaurants for the distance too might not have given any restaurants at all.

6 Discussion

Overall, our program's results demonstrate our goal's achievement. We managed to create a program that provides its users with recommended restaurants based on their input preferences and reviews from mutual users. The program, as anticipated, returns a visual representation of the recommended restaurants, their distance from the user and their cuisine.

We faced an issue with the longitudes and latitudes of the restaurants. The pre-modified Geospatial_Coordinates.csv file we used contained longitudes and latitudes of restaurants in Mexico. There were no data sets for longitudes and latitudes in Toronto and thus, we changed the postal codes to be postal codes in Toronto and used their corresponding longitudes and latitudes. However, we kept the restaurant names the same - names of restaurants in Mexico - as there were no data sets with Toronto restaurants that have been rated by the users in our user dataset. Hence, the restaurant names returned in the program being Spanish names. So, the modification we made was adding Toronto postal codes and longitudes and latitudes next to each of the restaurants to make the distance calculation realistic and within our target area.

One of our initial goals was to make 'preferred cuisines' one of the preference selections the program user could make but due to our short data set, adding an additional filter would have created the risk of not finding any cuisines with the required characteristics. Therefore, we decided to remove this filter to ensure we had enough restaurants to return to the user. Our algorithm/user similarity calculation, discussed in the computational part, ensured we got enough similar users back to then apply the rest of the program on.

Py-ta errors

We came across two py-ta errors. The first error was in the _WeightedVertex class where the parameter item was overriding the parameter kind. We added the error's code to the 'disable' section under config arguments in the py-ta block at the bottom of the recommender_data file since this was code we copied and did not modify from the exercise.

The second error was a "too many parameters" error in the User class. Since all the parameters were required for our program, we added this error's code to the 'disable' section under config arguments in the py-ta block at the bottom of the recommender_data file.

Finally, we came across a "future warning" error (not in py-ta). The error had to do with pandas which we installed to use plotly. The error does not affect any code and we did not know how to silence the warning.

Next steps for further exploration

Some other features that we can add to our program is filtering based on distance and preferred cuisine. One of the questions would be "What is your preferred cuisine?". The returned recommended restaurants would then only be restaurants that serve that specified cuisine. Filtering based on distance would first ask the user how far they want the restaurant to be from their specified location. It would then check the calculated distance between the user and each recommended restaurant and then only return the restaurant that is within the range of the inputted distance.

In conclusion, this program allowed us to create a restaurant recommender system for restaurants in Toronto but in a different way. Finding similar users allowed us to ensure our program user would be satisfied with the returned restaurants thus, making this program not only creative but effective.

7 References

UCI Machine Learning. “Restaurant Data with Consumer Ratings.” Kaggle.com, 2017, www.kaggle.com/datasets/uciml/restaurant-data-with-consumer-ratings.