SI507 Final Project Submission Document

Hongyu Dai

April 27, 2021

Project Code

Link to GitHub repo: https://github.com/hongyv0627/Final_Project.git

The README file can be checked in the GitHub Repository. The text is also provided below.

Special Instructions for running my code:

Before running the code, you need to apply API keys for Google Places API as well as Yelp Fusion API, put them in a python file named secrets.py as variables, and name the variables as GOOGLE_API_KEY and YELP_API_KEY respectively. In order to help SI507 Professors and GSIs easier to check my code, I will upload the secrets.py along with my final project submission pdf file when I submit the Final Project Submission assignment in Canvas.

A brief description of how to interact with my program:

The program is designated to check the google ratings and yelp ratings of a specific fast food brand's restaurants in a specific city. Users can enter their interested fast food brand name and enter their interested city name. The program will return most of (but not all) the restaurants' google rating and yelp rating of the brand in that city. The ratings data will be displayed in different kinds of plots, including histograms and scatter plots, and users can choose which plot do they want to look at.

Required Python packages for your program to successfully run: BeautifulSoup, requests, secrets, time, sqlite3, plotly.

Data sources

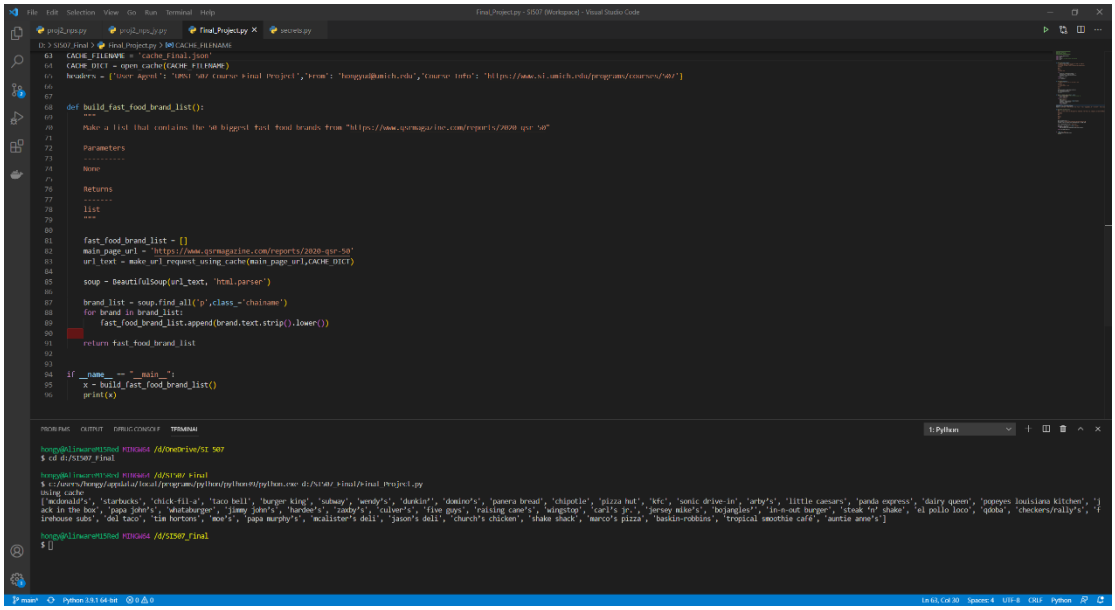
I use two APIs and one website for collecting data sources.

The first website is QSR50 ranking:

https://www.qsrmagazine.com/reports/2020-qsr-50

This ranking lists the 50 biggest fast-food brands in United States in 2020. I use web scraping to access the data, and also caching is used. In practice, users will only be allowed to search one of the 50 brands for their Google and Yelp ratings.

Summary of Data as well as evidence of caching:

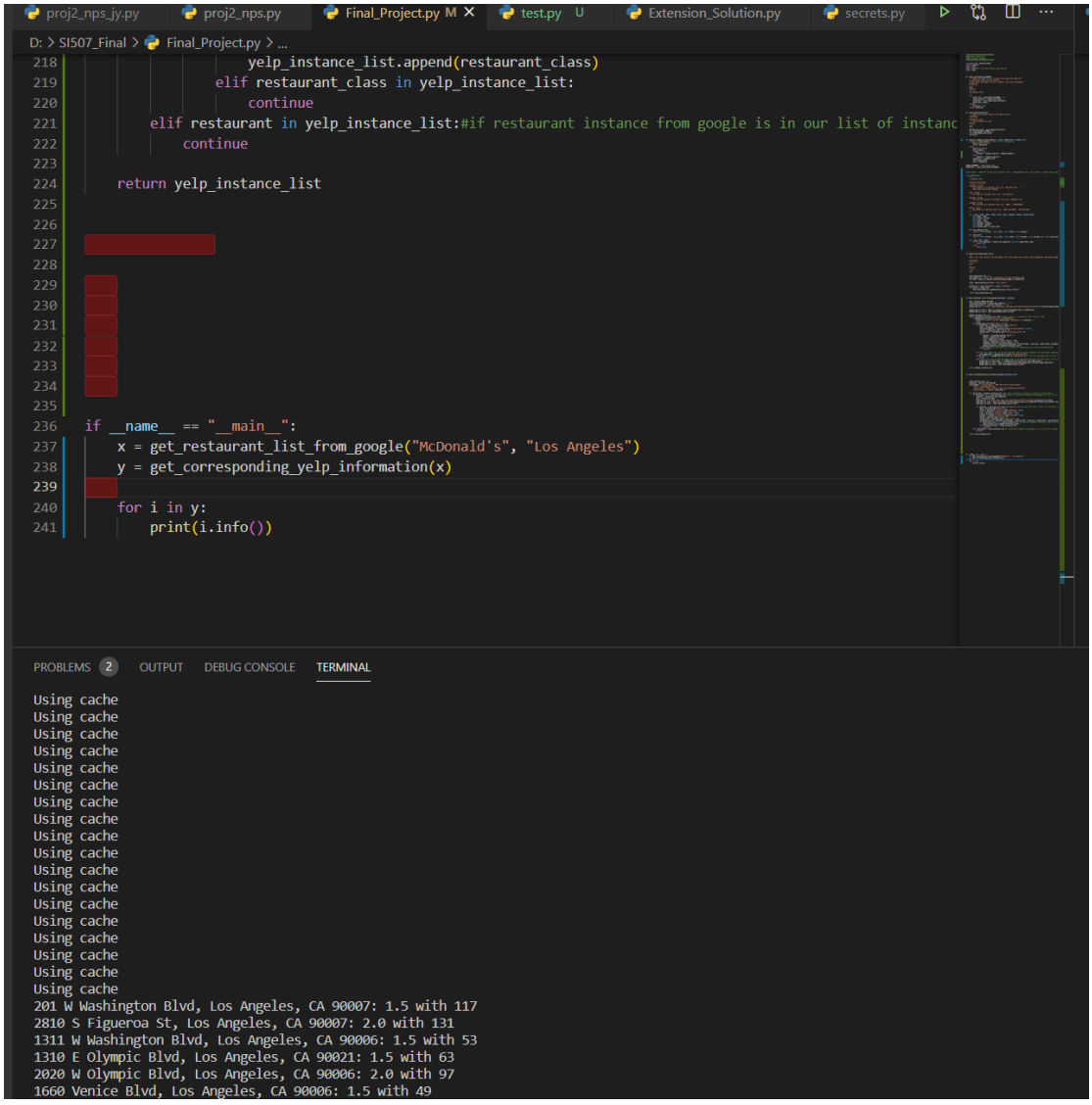


The first API is Google Places API: https://developers.google.com/maps/documentation/places/web-service/overview . The format is in JSON. I use API key to access the data, and also caching is used. For the records retrieved, I set a maximum number of 100, which means at most the API can return 100 records of restaurant. In practice, I try to search the McDonald's in Los Angeles (in the program the brand name and location will be the user's input), and google places API returns 60 records of restaurant, which is good. I choose the following fields: name; address, including street address, city, state, and zip code; rating, and total number of reviews.

The second API is Yelp fusion API: https://www.yelp.com/developers/documentation/v3/get_started . I use the name and address information from the result of google places API to search the restaurant's corresponding yelp rating and total number of reviews. I use API key to access the data, and also caching is used. In our case, foreign key would be name and address. Name and address links our two SQL tables. The fields are the same as Google Places API: name; address, including street address, city, state, and zip code; rating, and total number of reviews. In practice, when I pass the information of the 60 records from

Google Places API as input into the Yelps Fusion API, Yelps returns 191 records. This is usual because usually the search engine will use fuzzy search and the returned data will only partly satisfy our input information, which means we need to filter and match the restaurants afterwards to ensure that only the restaurants with both google ratings and yelp ratings provided can be left.

Below is the screenshot of caching: x is the function we get restaurants object from Google Places API, and y is the function we get restaurants object from Yelp Fusion API.



Database

As I said before, I use the restaurant's name and address that I get from Google Places API as my search term of Yelp Fusion API. Therefore, name and address will be the foreign key that links our two table. Primary key is just an auto-incremented ID, which has no relationship to our foreign key. In the program, I use two functions create_new_table and insert_new_value to create two tables, with each tables having the restaurants' name and address information as well as its Google/Yelp ratings and Google/Yelp review count, respectively.

Below is the screen shot of Google table and Yelp table:

Google Table:

	ID	Name	Street	City	State	Zipcode	Google_rating	Google_review_count
	过滤	过滤	过滤	过滤	过滤	过滤	过滤	过滤
1	1	McDonald's	201 W Washington Blvd	Los Angeles	CA	90007	2.7	5295
2	2	McDonald's	690 Alameda St	Los Angeles	CA	90021	3.7	2082
3	3	McDonald's	2810 S Figueroa St	Los Angeles	CA	90007	3.9	2459
4	4	McDonald's	1311 W Washington Blvd	Los Angeles	CA	90006	3.7	1480
5	5	McDonald's	1763 W Century Blvd	Los Angeles	CA	90047	3.6	1025
6	6	McDonald's	2020 W Olympic Blvd	Los Angeles	CA	90006	3.6	1862
7	7	McDonald's	1845 S La Cienega Blvd	Los Angeles	CA	90035	3.6	1349
8	8	McDonald's	5223 W Century Blvd	Los Angeles	CA	90045	3.7	1798
9	9	McDonald's	1310 E Olympic Blvd	Los Angeles	CA	90021	3.6	1985
10	10	McDonald's	6904 La Tijera Blvd	Los Angeles	CA	90045	3.9	1158
11	11	McDonald's	4166 Melrose Ave	Los Angeles	CA	90029	3.8	2243
12	12	McDonald's	3124 N San Fernando Rd	Los Angeles	CA	90065	3.6	1465
13	13	McDonald's	11300 National Blvd	Los Angeles	CA	90064	3.8	1042
14	14	McDonald's	1716 Marengo St	Los Angeles	CA	90033	3.9	2260
15	15	McDonald's	1210 S Soto St	Los Angeles	CA	90023	3.8	1111
16	16	McDonald's	10011 S Avalon Blvd	Los Angeles	CA	90003	3.8	1861
17	17	McDonald's	1160 Rosecrans Ave	Los Angeles	CA	90059	3.8	1341
18	18	McDonald's	5930 W Pico Blvd	Los Angeles	CA	90035	4	890
19	19	McDonald's	1007 N Western Ave	Los Angeles	CA	90029	3.7	1657
20	20	McDonald's	3602 South La Brea Ave	Los Angeles	CA	90016	3.7	1098
21	21	McDonald's	3501 S La Cienega Blvd	Los Angeles	CA	90016	3.6	1311
22	22	McDonald's	5450 Sunset Blvd	Los Angeles	CA	90027	3.7	1607
23	23	McDonald's	1625 Wilshire Blvd	Los Angeles	CA	90017	3.7	2193
24	24	McDonald's	341 S Vermont Ave	Los Angeles	CA	90020	3.7	2477

Yelp Table:

	ID	Name	Street	City	State	Zipcode	Yelp_rating	Yelp_review_count
	过滤	过滤	过滤	过滤	过滤	过滤	过滤	过滤
1	1	McDonald's	201 W Washington Blvd	Los Angeles	CA	90007	1.5	117
2	2	McDonald's	2810 S Figueroa St	Los Angeles	CA	90007	2.0	131
3	3	McDonald's	1311 W Washington Blvd	Los Angeles	CA	90006	1.5	53
4	4	McDonald's	1310 E Olympic Blvd	Los Angeles	CA	90021	1.5	63
5	5	McDonald's	2020 W Olympic Blvd	Los Angeles	CA	90006	2.0	97
6	6	McDonald's	1660 Venice Blvd	Los Angeles	CA	90006	1.5	49
7	7	McDonald's	4011 S Central	Los Angeles	CA	90011	1.0	43
8	8	McDonald's	1625 Wilshire Blvd	Los Angeles	CA	90017	2.0	120
9	9	McDonald's	690 Alameda St	Los Angeles	CA	90021	2.0	99
10	10	McDonald's	4000 S Figueroa St	Los Angeles	CA	90037	1.5	50
11	11	McDonald's	1071 Martin Luther King Jr Blvd	Los Angeles	CA	90037	1.5	66
12	12	McDonald's	1800 S Western Ave	Los Angeles	CA	90006	1.5	85
13	13	McDonald's	405 N Alvarado St	Los Angeles	CA	90026	2.0	83
14	14	McDonald's	1210 S Soto St	Los Angeles	CA	90023	1.5	33
15	15	McDonald's	341 S Vermont Ave	Los Angeles	CA	90020	1.5	171
16	16	McDonald's	1118 Slauson Ave	Los Angeles	CA	90011	1.5	92
17	17	McDonald's	695 S Western Ave	Los Angeles	CA	90005	2.0	200
18	18	McDonald's	3737 Soto St	Vernon	CA	90058	2.5	31
19	19	McDonald's	2215 W Martin Luther King Jr Blvd	Los Angeles	CA	90008	1.5	47
20	20	McDonald's	988 W Slauson Ave	Los Angeles	CA	90044	1.5	36
21	21	McDonald's	1763 W Century Blvd	Los Angeles	CA	90047	2.0	37
22	22	McDonald's	1406 W Manchester Ave	Los Angeles	CA	90047	1.5	37
23	23	McDonald's	2900 Imperial Hwy	Inglewood	CA	90303	1.5	77
24	24	McDonald's	501 W Imperial Hwy	Los Angeles	CA	90044	1.5	98

Then, I use the foreign keys (name, street, zip code) to match the restaurants and create a list of tuples that contains all the information we need. Finally, I use list and tuple indexing to extract the information and store in a dictionary of lists. Therefore, this dictionary of lists is our final data source, which we will use to generate our plots. Detailed explanation of this part will be provided in Demo video.

Interaction and Presentation Options

When the project is finished, users can input their interested fast food brand name and their interested city name. However, their brand name will be restricted to 50 biggest fast-food brands. After they input brand name, such as McDonald's, and city name, such as Los Angeles, the project will use google places API to help them find decades of McDonald's (maximum 100) in Los Angeles Area, with each McDonalds' rating and number of reviews being provided. Then, using Yelp Fusion API, each McDonald that we get from Google Places API will get their corresponding Yelp rating and number of reviews. In the program, the program will tell the users how many relevant results it found on Google how many relevant results it found on Yelp, as well as how many results left after matching. Note that the number of match results might be smaller than all of those two individual results. This is because the restaurant that we choose to examine is based on the results from google, and when we pass a specific restaurant's information that we get from Google Places API as input into Yelp Fusion API, it is possible that all of the results that we get from Yelp is not the information of that restaurant. In this case this restaurant will be dropped from our data source after matching.

After getting data, users can choose different datasets to plot different kinds of plots. There are 8 option of plots for them to choose to examine, which are:

1. Distribution of Google Rating
2. Distribution of Yelp Rating
3. Distribution of Weighted Rating
4. Scatter plot of Google Rating versus its Rating Numbers
5. Scatter plot of Yelp Rating versus its Rating Numbers
6. Scatter plot of Google and Yelp's Weighted Rating versus its total Rating Numbers
7. Scatter plot of Google Rating versus its Yelp Rating
8. Scatter plot of Google Rating number versus its Yelp Rating Numbers

Users can type the number and they will get corresponding plots.