Alexander Cotignola and Uddhav Bhagat

27 April 2020

SI 206 Final Project Report

Github Repository Link: https://github.com/uddhav99/SI-206-Final-Project

1. **Goals for this Project**
    a. Our main goal for this project was to do a search on both Yelp and Zomato for the 100 most popular restaurants across New York City and then compare the results. We wanted to collect name, address, latitude, longitude, price, rating (for both Yelp and Zomato), and review count (for both Yelp and Zomato). Using these data, we wanted to see how many of the same restaurants were in each search; see what the differences were in the price and rating across Yelp and Zomato for the same restaurant; see how many different cuisines were in each search; compare the review count on Yelp to the review count on Zomato (were the most-reviewed restaurants on Yelp also the most reviewed on Zomato?); and see how much the price of the restaurant correlated with review count on both Yelp and Zomato. Finally, we wanted to make a scatter map of the results from the search that colored each dot on the map based on price or rating.
2. **Goals We Achieved**
    a. We were able to gather all of the data we wanted from Yelp and Zomato from the categories listed above. Using a database join, we were able to see which restaurants appeared in both searches, and compared their ratings and prices on a line graph (plotting the Yelp price/rating, the Zomato price/rating, and the average price/rating for each restaurant as three separate lines). We were able to

compute the average price by adding the results from each column together and dividing by 2. By making a scatter plot, we were able to determine if review count on Yelp were positively correlated with review count on Zomato. We also made a graph plotting review count vs. price on both Yelp and Zomato (although it is hard to see correlation because the prices are discrete values). Finally, we were able to make two scatter maps, one for price and one for rating, that shows on a geographic map each restaurant and its color based on one of the criteria above.

3. **Problems We Faced**
    a. We were not able to see how many restaurants were in each category for Yelp and Zomato because the category names were not exactly the same, and so it would have been very difficult to compare any count of categories across the platforms. In addition, restaurants did not have exactly the same name across platforms, and therefore doing a database join on restaurant name might have missed some of the overlap between the searches. The addresses and latitude and longitude were also not formatted the same, so we could not join based on these criteria either. Originally, we were going to compare across a few different landmarks in the city, or by neighborhood, but this was rendered impossible by the fact that Zomato seemingly ignored any parameter to search within a specific radius, and so instead we got results all across the city. Also, once we did that specific search, the Zomato API did not give any new results, so we had to reframe our question around the results Zomato gave us. Given that the Yelp API did work for this, we had to reconfigure our Yelp search so that it would capture restaurants across a wider area than it previously did. We then adapted our

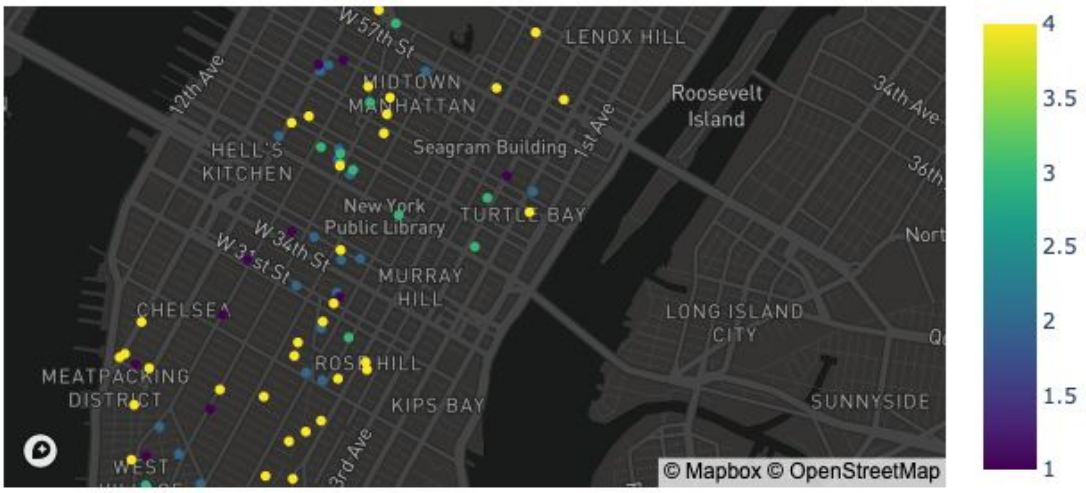
visualizations to compare the most popular restaurants across NYC, instead of a neighborhood like Chelsea or Greenwich Village.

4.

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | restaurant_name | yelp_price | yelp_rating | yelp_reviews | zomato_price | zomato_rating | zomato_reviews | average_price | average_rating |
| 2 | Grimaldi's Pizzeria | 2 | 3.5 | 4601 | 3 | 4.6 | 206 | 2.5 | 4.05 |
| 3 | Morimoto | 4 | 4 | 3055 | 4 | 4.4 | 88 | 4 | 4.2 |
| 4 | Minetta Tavern | 3 | 4 | 2188 | 4 | 4.3 | 71 | 3.5 | 4.15 |
| 5 | Beauty & Essex | 3 | 4 | 3495 | 4 | 4.3 | 63 | 3.5 | 4.15 |
| 6 | Sushi Yasuda | 3 | 4 | 2119 | 4 | 4.1 | 52 | 3.5 | 4.05 |
| 7 | Marea | 3 | 4 | 2034 | 4 | 4.2 | 73 | 3.5 | 4.1 |
| 8 | Scarpetta | 3 | 4 | 1904 | 4 | 4.2 | 43 | 3.5 | 4.1 |
| 9 | Carmine's Italian Restaurant - Times Square | 2 | 4 | 3779 | 4 | 4.7 | 161 | 3 | 4.35 |
| 10 | The Halal Guys | 1 | 4 | 9437 | 2 | 4.9 | 201 | 1.5 | 4.45 |
| 11 | Roberta's | 2 | 4 | 2837 | 3 | 4.8 | 71 | 2.5 | 4.4 |
| 12 | Joe's Shanghai | 2 | 4 | 6160 | 2 | 4.5 | 93 | 2 | 4.25 |
| 13 | Shake Shack | 2 | 4 | 5576 | 2 | 4.9 | 115 | 2 | 4.45 |
| 14 | Clinton Street Baking Company | 2 | 4 | 5063 | 4 | 4.4 | 98 | 3 | 4.2 |
| 15 | Katz's Delicatessen | 2 | 4 | 12205 | 2 | 4.9 | 538 | 2 | 4.45 |
| 16 | Ess-a-Bagel | 1 | 4 | 3582 | 1 | 4.7 | 82 | 1 | 4.35 |
| 17 | Mamoun's Falafel | 1 | 4 | 2384 | 2 | 4.6 | 45 | 1.5 | 4.3 |
| 18 | John's of Bleecker Street | 2 | 4 | 2016 | 3 | 4.6 | 62 | 2.5 | 4.3 |
| 19 | Gramercy Tavern | 4 | 4.5 | 2892 | 4 | 4.5 | 121 | 4 | 4.5 |
| 20 | Daniel | 4 | 4.5 | 1585 | 4 | 4.3 | 82 | 4 | 4.4 |
| 21 | Del Frisco's Double Eagle Steakhouse | 4 | 4.5 | 2977 | 4 | 4.5 | 54 | 4 | 4.5 |
| 22 | Jean-Georges | 4 | 4.5 | 1879 | 4 | 4.4 | 89 | 4 | 4.45 |

## 5. Visualizations

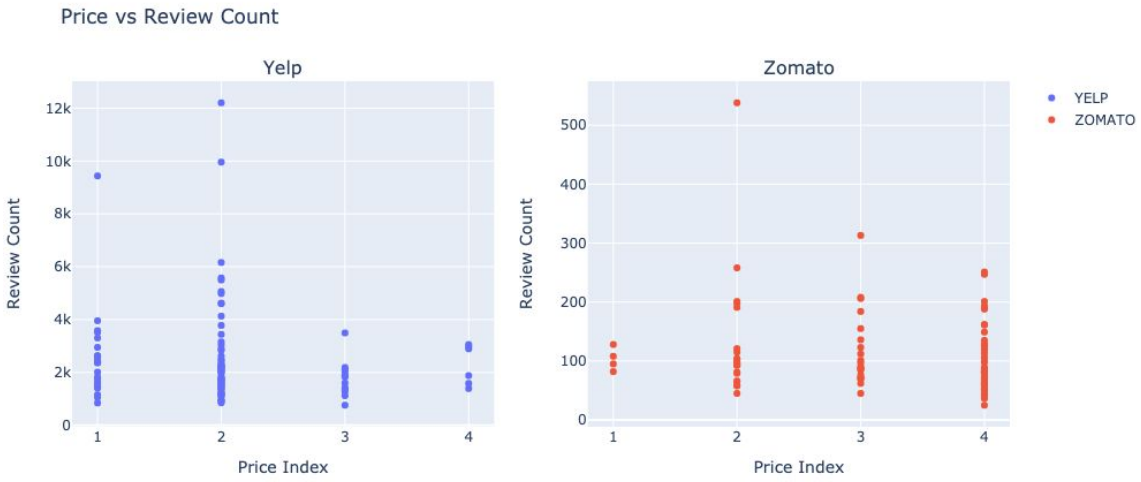
Distribution of restaurants, colored by price index




a.

b.


RATING COMPARISON
Yelp Rating
Zomato Rating
Average Rating
Rating Index
5
4.5
4
3.5
3
Grimaldi's Pizzeria
Morimoto
Minetta Tavern
Beauty & Essex
Sushi Yasuda
Marea
Scarpetta
Carmine's Italian Restaurant - Times Square
The Halal Guys
Roberta's
Joe's Shanghai
Shake Shack
Clinton Street Baking Company
Katz's Delicatessen
Ess-a-Bagel
Mamoun's Falafel
John's of Bleecker Street
Gramercy Tavern
Daniel
Del Frisco's Double Eagle Steakhouse
Jean-Georges
Restaurant Name




c.


d.

## 6. Instructions

a. When you run yelp_api.py, you will be prompted to enter an offset. Enter a multiple of 20 (starting from 0) as a number using your keyboard. This allows you to get different results for each search. Further, the location parameter must be a

string. Make sure that the restaurants.sqlite database is in the same folder (or you specify a path); otherwise, you won't be able to connect to the database. You must also have an internet connection, since the requests module fetches live results from the Yelp API.

b. When you run zomato_api.py, you will be prompted to enter an offset. Enter a multiple of 20 (starting from 0) as a number using your keyboard. This allows you to get different results for each search. Further, the latitude and longitude must be in the form of a float, with a dash (negative sign) indicating south or west for latitude and longitude, respectively. Make sure that the restaurants.sqlite database is in the same folder (or you specify a path); otherwise, you won't be able to connect to the database. You must also have an internet connection, since the requests module fetches live results from the Zomato API.

c. To run visualisations_uddhav.py, you don't need to input any values into the console. Make sure that the restaurants.sqlite database is in the same folder (or you specify a path); otherwise, you won't be able to connect to the database. You must also have an internet connection, since the code makes the visualizations in your browser. The code will automatically open a tab for each visualization in your default browser window, and you can click through each visualization as desired. To access the plotly toolbar for interacting with the visualization, hover over the top of the page, and a set of options should appear. You can download the plot as a png file (which will default to putting it in your Downloads folder), as well as select data or zoom in as you prefer. You can also hover over each dot on the visualization, and it will give you relevant information, such as restaurant name and rating.

7. **Documentation**

    a. **yelp_api.py**

        i. def setUpDatabase(data, conn, cur)

            1. This function takes in a dictionary of Yelp restaurant data, sqlite connection, and sqlite cursor. It then parses through each restaurant to grab relevant information for each column in the database, and creates a new table called Popular_Restaurants in the restaurants.sqlite with each row being a restaurant and each column being a restaurant attribute (such as price, address, rating). It will ignore duplicate restaurants while adding. The connection will then commit these changes. There is no return value.

        ii. def getCategories(data, conn, cur)

            1. This function takes in a dictionary of Yelp restaurant data, a sqlite connection, and a sqlite cursor. It then loops through each restaurant in the dictionary, captures its category, and sees if that restaurant's category is in the list of categories; if not, it adds that category to the list of categories. It then creates a Categories_Yelp table in the restaurants.sqlite database, with an id as the primary key and the category as text. Then, it loops through the category list and defines an id for each category. It will ignore any duplicate categories since there is a UNIQUE keyword in the creation of the table. The connection will then commit these changes. There is no return value.

iii. def getData(location,offset)

1. This function takes in a search term to use for Yelp search and an offset to determine which 20 results the Yelp API returns. It first connects to the restaurants.sqlite database and sets up a sqlite cursor. We use the Yelp API key, the search term, the category type, and the offset to form a URL that is then inputted into the requests.get function. The json that is returned is then converted into a dictionary called data using json.loads(). This function then calls getCategories and setUpDatabase (using the data dictionary, connection, and cursor from above) to add the categories and restaurants to their respective tables in the restaurants.sqlite database. If the URL is invalid, the dictionary is then set to be empty. The fetching then pauses for 5 seconds to ensure we don't get rate limited by the Yelp API. There is no return value.

**b. zomato_api.py**

i. def set_table(data, conn, cur)

1. This function takes in a dictionary of Zomato restaurant data, a sqlite connection, and a sqlite cursor. It then parses through each restaurant to grab relevant information for each column in the database, and creates a new table called Zomato_Restaurants in the restaurants.sqlite with each row being a restaurant and each column being a restaurant attribute (such as price, address, rating, etc.). It will ignore duplicate restaurants while adding. The

connection will then commit these changes. There is no return value.

ii. def getCategories(data, conn, cur)

1. This function takes in a dictionary of Zomato restaurant data, sqlite connection, and sqlite cursor. It then loops through each restaurant in the dictionary, captures its category, and sees if that restaurant's category is in the list of categories; if not, it adds that category to the list of categories. It then creates a Categories_Zomato table in the restaurants.sqlite database, with an id as the primary key and the category as text. Then, it loops through the category list and defines an id for each category. It will ignore any duplicate categories since there is a UNIQUE keyword in the creation of the table. The connection will then commit these changes. There is no return value.

iii. def get_data(latitude,longitude,offset)

1. This function takes in a latitude and longitude to use for Zomato search and an offset to determine which 20 results the Zomato API returns. It first connects to the restaurants.sqlite database and sets up a sqlite cursor. We use the Zomato API key, the latitude, the longitude, and the offset to form a URL that is then inputted into the requests.get function. The json that is returned is then converted into a dictionary called data using json.loads(). This function then calls getCategories and set_table (using the data dictionary, connection, and cursor from above) to add the

categories and restaurants to their respective tables in the restaurants.sqlite database. The fetching then pauses for 5 seconds to ensure we don't get rate limited by the Zomato API. There is no return value.

**c. visualisations_uddhav.py**

i. def join(conn, cur):

1. This function takes in a sqlite connection and cursor. It then joins the Popular_Restaurants and Zomato_Restaurants tables based on restaurant name, getting a list of names, prices, ratings, and review counts for each restaurant on both Yelp and Zomato. It then puts these values into a Pandas dataframe, dropping any duplicates (to account for chains with the same name but a different address). Next, it sorts the yelp_price category in ascending order, and averages the Yelp and Zomato prices in a new column. After that, it adds three lines to a Plotly line graph figure object, with points for each restaurant: the Yelp price, the Zomato price, and the average price between them. We then make a Plotly scatterplot with the Yelp review count on the x-axis and the Zomato review count on the y-axis. The function then outputs these graphs to HTML files, and outputs the calculations from the dataframe to a CSV. There is no return value. After running this function, the line graph and scatter plot will open automatically in the browser.

ii. def scatter_plot(conn, cur)

1. This function takes in a sqlite connection and cursor and creates two side-by-side scatter plots using data from the restaurants.sqlite database. It selects the restaurant name, rating, and review count for both Yelp and Zomato, and puts these into two separate Pandas dataframes. Using Plotly, the function makes a scatterplot with Price on the x-axis and Review Count on the y-axis, and makes side-by-side scatter plots on the same page. The function then outputs these graphs to HTML files, and automatically opens a tab in your web browser. There is no return value.

iii. def chloropleth_maps(conn, cur):

1. This function takes in a sqlite connection and cursor and creates two scatter maps using data from the restaurants.sqlite database. Taking each restaurant's name, latitude, longitude, rating, and price, the function puts these values into Yelp and Zomato Pandas dataframes. It then concatenates the data using a pd.concatenate function, and removes any duplicate values to account for chains with different locations. Next, using the scatter.mapbox function from Plotly and a Mapbox API key, it creates a scatter map centered on New York City, one based on price and one based on rating. The color scale is based on either price or rating, depending on the map. The color of each dot indicates its rating or price, with a scale on the side for reference. When hovering over a dot, you can see the restaurant name and location. The function

then outputs these graphs to HTML files, and automatically opens a tab in your web browser. There is no return value.

8. **Documentation of Resources**

| Date | Issue Description | Location of Resource | Result (did it resolve the issue?) |
|---|---|---|---|
| 4-7-20 | How to set up URL for Yelp API request | https://python.gotrained.com/yelp-fusion-api-tutorial/ | Yes; I was able to figure out the parameters and headers for the request URL, including how to include the API Key |
| 4-8-20 | What parameters to include in Yelp API search, and how to format them | https://www.yelp.com/developers/documentation/v3/business_search | Yes; I was able to figure out which parameters to use |
| 4-11-20 | How to concatenate two dataframes using Pandas | https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.concat.html | Yes |
| 4-12-20 | What parameters to include in Zomato API search, and how to format them | https://developers.zomato.com/documentation#/ | Yes; I was able to figure out which parameters to use |
| 4-18-20 | How to create scatter plots using plotly | https://plotly.com/python/line-and-scatter/ | Yes |
| 4-18-20 | Using mapbox and making scatter maps using Plotly | https://plotly.com/python/scattermapbox/ | Yes |
| 4-18-20 | How to set labels in plotly figures | https://plotly.com/python/figure-labels/ | Yes |
| 4-19-20 | How to export a Pandas dataframe to CSV, and how to | https://pandas.pydata.org/pandas-docs/stable/reference/api | Yes |

| | | | |
|---|---|---|---|
| | not include row numbers | /pandas.DataFrame.to_csv.html | |