Compare the similarities of financial world cities (Manhattan and Toronto)

Jiaqing Chen
2020/08/23

1. Introduction

1.1. Background

A global city, also called a power city, world city, alpha city or world center, is a city which is a primary node in the global economic network. The concept comes from geography and urban studies, and the idea that globalization is created and furthered in strategic geographic locales according to a hierarchy of importance to the operation of the global system of finance and trade. A large city carries the development opportunities of a region. A world-class city is a place for investors and job seekers alike. There are many common characteristics between these cities. Looking for these characteristics not only helps us to analyze the future development of big cities, but also helps us to measure which city has these common parts and will develop into a big city in the future. This is a very meaningful thing for many people, especially investors. This will help trend the direction of capital development.

1.2. Problem

In part of the previous work, I explored New York City and the city of Toronto and segmented and clustered their neighborhoods. Both cities are very diverse and are the financial capitals of their respective countries. One interesting idea is to compare the neighborhoods of the two cities and determine how similar they are. The problem the project is trying to solve is in which aspects and to what extent the two cities are similar. Through comparison and analysis, the project attempts to draw a conclusion in which aspects can we see whether a city has the potential to develop into a large financial city.

2. Data

2.1. Description

The datasets needed to complete this project are what I used in previous work. They include the data from New York and Toronto, respectively. For the New York dataset, the Neighborhood has a total of 5 boroughs and 306 neighborhoods. In order to

segment the neighborhoods and explore them, I essentially need a dataset that contains the 5 boroughs and the neighborhoods that exist in each borough as well as the latitude and longitude coordinates of each neighborhood. The dataset exists for free on the web and the link to the dataset is shown as: https://geo.nyu.edu/catalog/nyu_2451_34572. For the Toronto neighborhood data, a Wikipedia page exists that has all the information we need to explore and cluster the neighborhoods in Toronto. In general, it is important to first scrape the Wikipedia page and wrangle the data, clean it, and then read it into a pandas dataframe so that it is in a structured format like the New York dataset. The link to the Wikipedia page of the dataset is shown as:
https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M.

2.2. Usage Method

These two datasets are first faced with some cleaning work. After downloading the data, the first work is to transform the data into a panda dataframe. Next, we need to use some geopy libraries to get the latitude and longitude values of the two cities and create two maps of them with neighborhoods superimposed on top. Here, we need to use a library called Folium, which is a great visualization library. We could zoom into the above map, and click on each circle mark to reveal the name of the neighborhood and its respective borough. Next, we are going to start utilizing the Foursquare API to explore the neighborhoods and segment them. After completing the above steps, we will face the most important part, analysis work. We need group rows by neighbor and by taking the mean of the frequency of occurrence of each category. Finally, we use the k-means method for clustering neighborhoods. This includes the work of comparison and analysis. This project would exchange test datasets to determine how similar the two models are. The exchanged test datasets are compared with the non-exchanged test datasets. At the same time, the two models are compared intuitively.

3. Methodology

3.1. Overview

The whole data analysis and comparison section covers data download, cleaning, merging and analysis. As the project is mainly for data analysis of the two regions, the analysis steps and processes are basically similar. The difference between them is mainly the data acquisition part. One data is downloaded from Wikipedia. The processing of this data is complicated. The other data is loaded directly in the form of CSV. This project is also designed to show two different ways to load data. After the data is loaded, the project cleans up and merges the data appropriately. This process clearly sorts out the data. Next, the data is analyzed step by step, and the results of a single regional data set are obtained. Finally, the results of the two regions are compared and the analysis report is output. In the process of data analysis, this project uses the Foursquare library. This will be described later in the report.

3.2. Data Collation and Analysis

For the data download and collation part, this project will describe the two methods adopted by itself. The full version of this code can be found in the link to the project. In the method of downloading by link, the data must be imported into the dataframe through the link, which is as shown below.

Scrape the List of postal codes of Canada from URL

In [2]:
```
ca_url = "https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M"
ca_source = requests.get(ca_url).text
ca_soup = BeautifulSoup(ca_source, 'xml')
ca_table=ca_soup.find('table')
ca_column_names = ['Postalcode','Borough','Neighborhood']
ca_df = pd.DataFrame(columns = ca_column_names)
```

Next, the data needs to be further cleaned by searching and sorting. The operation of each step is shown in the following process.

In [4]:
```
for tr_cell in ca_table.find_all('tr'):
    row_data=[]
    for td_cell in tr_cell.find_all('td'):
        row_data.append(td_cell.text.strip())
    if len(row_data)==3:
        ca_df.loc[len(ca_df)] = row_data
```

In [5]:
```
ca_df.head()
```

Out[5]:

	Postalcode	Borough	Neighborhood
0	M1A	Not assigned	Not assigned
1	M2A	Not assigned	Not assigned
2	M3A	North York	Parkwoods
3	M4A	North York	Victoria Village
4	M5A	Downtown Toronto	Regent Park, Harbourfront

In [6]:
```
ca_df=ca_df[ca_df['Borough']!='Not assigned']
```

In [7]:
```
ca_df.head()
```

Out[7]:

	Postalcode	Borough	Neighborhood
2	M3A	North York	Parkwoods
3	M4A	North York	Victoria Village
4	M5A	Downtown Toronto	Regent Park, Harbourfront
5	M6A	North York	Lawrence Manor, Lawrence Heights
6	M7A	Downtown Toronto	Queen's Park, Ontario Provincial Government

Finally, we will get the following results through the analysis of this part.

In [8]:
```
ca_df.Neighborhood.replace('Not assigned',ca_df.Borough,inplace=True)
ca_df.head()
```

Out[8]:

	Postalcode	Borough	Neighborhood
2	M3A	North York	Parkwoods
3	M4A	North York	Victoria Village
4	M5A	Downtown Toronto	Regent Park, Harbourfront
5	M6A	North York	Lawrence Manor, Lawrence Heights
6	M7A	Downtown Toronto	Queen's Park, Ontario Provincial Government

In [9]:
```
ca_df = ca_df.groupby(['Postalcode', 'Borough'])['Neighborhood'].apply(lambda x: ', '.join(x))
ca_df = ca_df.reset_index()
ca_df.rename(columns = {'Postalcode':'PostalCode'}, inplace = True)
ca_df.rename(columns = {'Neighborhood':'Neighborhood'}, inplace = True)
ca_df.head()
```

Out[9]:

	PostalCode	Borough	Neighborhood
0	M1B	Scarborough	Malvern, Rouge
1	M1C	Scarborough	Rouge Hill, Port Union, Highland Creek
2	M1E	Scarborough	Guildwood, Morningside, West Hill
3	M1G	Scarborough	Woburn
4	M1H	Scarborough	Cedarbrae

At this point, we need to download the data from the new link and merge the two parts of the data. Before merging two parts of data, we should make sure that the two parts of data can be properly merged.

In [11]:
```
ca_geo_url = 'http://cocl.us/Geospatial_data'
ca_df_geo=pd.read_csv(ca_geo_url)
ca_df_geo.head()
```

Out[11]:

	Postal Code	Latitude	Longitude
0	M1B	43.806686	-79.194353
1	M1C	43.784535	-79.160497
2	M1E	43.763573	-79.188711
3	M1G	43.770992	-79.216917
4	M1H	43.773136	-79.239476

In [12]:
```
ca_df_geo.shape
```

Out[12]: (103, 3)

In [13]:
```
ca_df_geo.rename(columns={'Postal Code':'PostalCode'},inplace=True)
ca_geo_merged = pd.merge(ca_df_geo, ca_df, on='PostalCode')
ca_geo_data=ca_geo_merged[['PostalCode','Borough','Neighborhood','Latitude','Longitude']]
ca_geo_data.head()
```

Out[13]:

	PostalCode	Borough	Neighborhood	Latitude	Longitude
0	M1B	Scarborough	Malvern, Rouge	43.806686	-79.194353
1	M1C	Scarborough	Rouge Hill, Port Union, Highland Creek	43.784535	-79.160497
2	M1E	Scarborough	Guildwood, Morningside, West Hill	43.763573	-79.188711
3	M1G	Scarborough	Woburn	43.770992	-79.216917
4	M1H	Scarborough	Cedarbrae	43.773136	-79.239476

Next, we will define a new function to get the top 100 venues that are in Toronto within a radius of 500 meters.

Get the top 100 venues that are in Toronto within a radius of 500 meters

In [24]:
```
def getNearbyVenues(names, latitudes, longitudes):
    radius=500
    LIMIT=100
    venues_list=[]
    for name, lat, lng in zip(names, latitudes, longitudes):
        print(name)

        # create the API request URL
        url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'.format(
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            lat,
            lng,
            radius,
            LIMIT)

        # make the GET request
        results = requests.get(url).json()["response"]['groups'][0]['items']

        # return only relevant information for each nearby venue
        venues_list.append([(
            name,
            lat,
            lng,
            v['venue']['name'],
            v['venue']['location']['lat'],
            v['venue']['location']['lng'],
            v['venue']['categories'][0]['name']) for v in results])

    nearby_venues = pd.DataFrame([item for venue_list in venues_list for item in venue_list])
    nearby_venues.columns = ['Neighborhood',
                  'Neighborhood Latitude',
                  'Neighborhood Longitude',
                  'Venue',
                  'Venue Latitude',
                  'Venue Longitude',
                  'Venue Category']

    return(nearby_venues)
```

Then, we will call this function to analyze and process the dataset and get new results.

```
In [25]: ca_toronto_venues = getNearbyVenues(names=ca_toronto_data['Neighborhood'],
                                    latitudes=ca_toronto_data['Latitude'],
                                    longitudes=ca_toronto_data['Longitude']
                                   )
```

```
The Beaches
The Danforth West, Riverdale
India Bazaar, The Beaches West
Studio District
Lawrence Park
Davisville North
North Toronto West,  Lawrence Park
Davisville
Moore Park, Summerhill East
Summerhill West, Rathnelly, South Hill, Forest Hill SE, Deer Park
Rosedale
St. James Town, Cabbagetown
Church and Wellesley
Regent Park, Harbourfront
Garden District, Ryerson
St. James Town
Berczy Park
Central Bay Street
Richmond, Adelaide, King
Harbourfront East, Union Station, Toronto Islands
Toronto Dominion Centre, Design Exchange
Commerce Court, Victoria Hotel
Roselawn
Forest Hill North & West, Forest Hill Road Park
The Annex, North Midtown, Yorkville
University of Toronto, Harbord
Kensington Market, Chinatown, Grange Park
CN Tower, King and Spadina, Railway Lands, Harbourfront West, Bathurst Quay, South Niagara, Island airport
Stn A PO Boxes
First Canadian Place, Underground city
Christie
Dufferin, Dovercourt Village
Little Portugal, Trinity
Brockton, Parkdale Village, Exhibition Place
High Park, The Junction South
Parkdale, Roncesvalles
Runnymede, Swansea
Queen's Park, Ontario Provincial Government
Business reply mail Processing Centre, South Central Letter Processing Plant Toronto
```

At this time, we need to sort out the data and get the counting results that are conducive to our further classification and sorting. Here, this report will not show this part of the steps one by one. The specific process is in the project link. Only the final statistics are shown here.

```
In [32]: ca_toronto_grouped = ca_toronto_onehot.groupby('Neighborhood').mean().reset_index()
         ca_toronto_grouped.head()
```

Out[32]:

	Neighborhood	Airport	Airport Food Court	Airport Gate	Airport Lounge	Airport Service	Airport Terminal	American Restaurant	Antique Shop	Aquarium	...	Theme Restaurant	Toy / Game Store	Trail	Train Station	Vegetarian / Vegan Restaurant
0	Berczy Park	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.017241
1	Brockton, Parkdale Village, Exhibition Place	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.000000
2	Business reply mail Processing Centre, South C...	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.000000
3	CN Tower, King and Spadina, Railway Lands, Har...	0.066667	0.066667	0.066667	0.133333	0.2	0.133333	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.000000
4	Central Bay Street	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.015385

5 rows × 240 columns

Through the above data, we further develop a new statistical function.

In [33]:
```
def return_most_common_venues(row, num_top_venues):
    row_categories = row.iloc[1:]
    row_categories_sorted = row_categories.sort_values(ascending=False)
    return row_categories_sorted.index.values[0:num_top_venues]
```

In [37]:
```
num_top_venues = 10

indicators = ['st', 'nd', 'rd']

# create columns according to number of top venues
columns = ['Neighborhood']
for ind in np.arange(num_top_venues):
    try:
        columns.append('{}{} Most Common Venue'.format(ind+1, indicators[ind]))
    except:
        columns.append('{}th Most Common Venue'.format(ind+1))

# create a new dataframe
neighborhoods_venues_sorted = pd.DataFrame(columns=columns)
neighborhoods_venues_sorted['Neighborhood'] = ca_toronto_grouped['Neighborhood']

for ind in np.arange(ca_toronto_grouped.shape[0]):
    neighborhoods_venues_sorted.iloc[ind, 1:] = return_most_common_venues(ca_toronto_grouped.iloc[ind, :], num_top_venues)

neighborhoods_venues_sorted.head()
```

By executing this part of the function, we get the final result of one of the datasets.

Out[49]:

Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
The Beaches	43.676357	-79.293031	0	Health Food Store	Pub	Trail	Dog Run	Dessert Shop	Dim Sum Restaurant	Diner	Discount Store	Distribution Center	Yoga Studio
The Danforth West, Riverdale	43.679557	-79.352188	0	Greek Restaurant	Coffee Shop	Italian Restaurant	Restaurant	Ice Cream Shop	Furniture / Home Store	Fruit & Vegetable Store	Pub	Pizza Place	Lounge
India Bazaar, The Beaches West	43.668999	-79.315572	0	Sushi Restaurant	Pub	Sandwich Place	Light Rail Station	Board Shop	Liquor Store	Burrito Place	Italian Restaurant	Restaurant	Ice Cream Shop
Studio District	43.659526	-79.340923	0	Café	Coffee Shop	Gastropub	Bakery	Brewery	American Restaurant	Yoga Studio	Convenience Store	Sandwich Place	Cheese Shop
Lawrence Park	43.728020	-79.388790	2	Park	Bus Line	Swim School	Department Store	Electronics Store	Eastern European Restaurant	Dumpling Restaurant	Donut Shop	Doner Restaurant	Dog Run

For another dataset, the steps we take will be simpler. The two files here are already saved and exist in the project directory. We first import the first batch of data.

Read csv file with clustered neighborhoods with geodata of Manhattan

In [15]:
```
manhattan_data  = pd.read_csv('mh_neigh_data.csv')
manhattan_data.head()
```

Out[15]:

	Borough	Neighborhood	Latitude	Longitude	Cluster Labels
0	Manhattan	Marble Hill	40.876551	-73.910660	2
1	Manhattan	Chinatown	40.715618	-73.994279	2
2	Manhattan	Washington Heights	40.851903	-73.936900	4
3	Manhattan	Inwood	40.867684	-73.921210	3
4	Manhattan	Hamilton Heights	40.823604	-73.949688	0

In [16]:
```
manhattan_data.tail()
```

Out[16]:

	Borough	Neighborhood	Latitude	Longitude	Cluster Labels
35	Manhattan	Turtle Bay	40.752042	-73.967708	3
36	Manhattan	Tudor City	40.746917	-73.971219	3
37	Manhattan	Stuyvesant Town	40.731000	-73.974052	4
38	Manhattan	Flatiron	40.739673	-73.990947	3
39	Manhattan	Hudson Yards	40.756658	-74.000111	2

Next, we can directly get the data results of this region by importing the second batch of data.

```
In  [17]: manhattan_merged = pd.read_csv('manhattan_merged.csv')
          manhattan_merged.head()
```

Out[17]:

	Borough	Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Com V
0	Manhattan	Marble Hill	40.876551	-73.910660	2	Coffee Shop	Discount Store	Yoga Studio	Steakhouse	Supplement Shop	Tennis Stadium	Shoe Store	Gym	
1	Manhattan	Chinatown	40.715618	-73.994279	2	Chinese Restaurant	Cocktail Bar	Dim Sum Restaurant	American Restaurant	Vietnamese Restaurant	Salon / Barbershop	Noodle House	Bakery	Bı Tea
2	Manhattan	Washington Heights	40.851903	-73.936900	4	Café	Bakery	Mobile Phone Shop	Pizza Place	Sandwich Place	Park	Gym	Latin American Restaurant	T Resta
3	Manhattan	Inwood	40.867684	-73.921210	3	Mexican Restaurant	Lounge	Pizza Place	Café	Wine Bar	Bakery	American Restaurant	Park	Fr Y
4	Manhattan	Hamilton Heights	40.823604	-73.949688	0	Mexican Restaurant	Coffee Shop	Café	Deli / Bodega	Pizza Place	Liquor Store	Indian Restaurant	Sushi Restaurant	Sanc I

After the two parts of data are loaded and sorted out, we will compare and analyze the two groups of data. We wanted to apply the data to maps and analyze them. However, through practice, I found that this can not get good results. On the contrary, it is more appropriate to analyze the data table directly.

3.3. Foursquare

As mentioned earlier, the library Foursquare is used in this project. This part is mainly used to segment and cluster the neighborhoods. Foursquare, which is short for Foursquare City Guide, is a local search-and-discovery mobile app developed by Foursquare Labs Inc. The app provides personalized recommendations of places to go near a user's current location based on users' previous browsing history and check-in history.

```
In  [19]: !conda install -c conda-forge geocoder --yes
          !conda install -c conda-forge geopy --yes
          !pip install lxml

          import geocoder
          from geopy.geocoders import Nominatim

          address = 'Toronto, Ontario'

          geolocator = Nominatim(user_agent="toronto_explorer")
          location = geolocator.geocode(address)
          latitude = location.latitude
          longitude = location.longitude

          Collecting package metadata (current_repodata.json): ...working... done
          Solving environment: ...working... done

          ## Package Plan ##

            environment location: D:\Anaconda

            added / updated specs:
              - geocoder


          The following packages will be downloaded:

              package                    |            build
              ---------------------------|-----------------
              conda-4.8.4                |   py38h32f6830_2         3.1 MB  conda-forge
              ------------------------------------------------------------
                                                     Total:         3.1 MB

          The following packages will be UPDATED:

            conda                              4.8.4-py38h32f6830_1 --> 4.8.4-py38h32f6830_2


          Downloading and Extracting Packages
```

The usage of this library will not be discussed here. This part of the content can be found in the course materials.

4. Results

Since the effect of map display is not as good as expected. And the map display can not clearly get the contrast results. Finally, this project adopts the method of direct comparison of tables, which are dataframes to get the final result. Here we will show the data analysis results of the two regions and the comparison results of the data analysis results of the two regions. The comparison results can not be fully displayed. Specific data run results can be found in the project link. The statistical results of the two regions are shown below.

Out[49]:

Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
The Beaches	43.676357	-79.293031	0	Health Food Store	Pub	Trail	Dog Run	Dessert Shop	Dim Sum Restaurant	Diner	Discount Store	Distribution Center	Yoga Studio
The Danforth West, Riverdale	43.679557	-79.352188	0	Greek Restaurant	Coffee Shop	Italian Restaurant	Restaurant	Ice Cream Shop	Furniture / Home Store	Fruit & Vegetable Store	Pub	Pizza Place	Lounge
India Bazaar, The Beaches West	43.668999	-79.315572	0	Sushi Restaurant	Pub	Sandwich Place	Light Rail Station	Board Shop	Liquor Store	Burrito Place	Italian Restaurant	Restaurant	Ice Cream Shop
Studio District	43.659526	-79.340923	0	Café	Coffee Shop	Gastropub	Bakery	Brewery	American Restaurant	Yoga Studio	Convenience Store	Sandwich Place	Cheese Shop
Lawrence Park	43.728020	-79.388790	2	Park	Bus Line	Swim School	Department Store	Electronics Store	Eastern European Restaurant	Dumpling Restaurant	Donut Shop	Doner Restaurant	Dog Run

In [50]:
```
manhattan_merged.head()
```

Out[50]:

Neighborhood	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue	10th Most Common Venue
Marble Hill	40.876551	-73.910660	2	Coffee Shop	Discount Store	Yoga Studio	Steakhouse	Supplement Shop	Tennis Stadium	Shoe Store	Gym	Bank	Seafood Restaurant
Chinatown	40.715618	-73.994279	2	Chinese Restaurant	Cocktail Bar	Dim Sum Restaurant	American Restaurant	Vietnamese Restaurant	Salon / Barbershop	Noodle House	Bakery	Bubble Tea Shop	Ice Cream Shop
Washington Heights	40.851903	-73.936900	4	Café	Bakery	Mobile Phone Shop	Pizza Place	Sandwich Place	Park	Gym	Latin American Restaurant	Tapas Restaurant	Mexican Restaurant
Inwood	40.867684	-73.921210	3	Mexican Restaurant	Lounge	Pizza Place	Café	Wine Bar	Bakery	American Restaurant	Park	Frozen Yogurt Shop	Spanish Restaurant
Hamilton Heights	40.823604	-73.949688	0	Mexican Restaurant	Coffee Shop	Café	Deli / Bodega	Pizza Place	Liquor Store	Indian Restaurant	Sushi Restaurant	Sandwich Place	Yoga Studio

The comparison results of the statistical results of the two regions are as follows.

Compare the information from the two regions

In [56]:
```
! pip install datacompy

import datacompy, pandas as pd, sys

compare = datacompy.Compare(toronto_merged, manhattan_merged, join_columns=['1st Most Common Venue','2nd Most Common Venue','3rd Most Comm
print(compare.matches())
print(compare.report())
```

```
28  Downtown Toronto                                       Stn A PO Boxes  43.646435 -79.374846        0.0
Coffee Shop              Pub                Café             Beer Bar                Restaurant
9   Central Toronto  Summerhill West, Rathnelly, South Hill, Forest Hill SE, Deer Park  43.686412 -79.400049        0.0
Coffee Shop              Pub             Bagel Shop           Supermarket          Vietnamese Restaurant
10  Downtown Toronto                                                 Rosedale  43.679563 -79.377529        2.0
Park               Trail              Playground          Dance Studio  Eastern European Restaurant
31      West Toronto                                  Dufferin, Dovercourt Village  43.669005 -79.442259        0.0
Pharmacy               Bakery          Grocery Store    Athletics & Sports          Gym / Fitness Center
3       East Toronto                                         Studio District  43.659526 -79.340923        0.0
Café          Coffee Shop             Gastropub                Bakery                 Brewery
23   Central Toronto                Forest Hill North & West, Forest Hill Road Park  43.696948 -79.411307        3.0
Jewelry Store                Trail    Mexican Restaurant       Sushi Restaurant                 Yoga Studio
33      West Toronto                    Brockton, Parkdale Village, Exhibition Place  43.636847 -79.428191        0.0
Café         Breakfast Spot           Coffee Shop           Yoga Studio                         Gym
13  Downtown Toronto                                    Regent Park, Harbourfront  43.654260 -79.360636        0.0
Coffee Shop             Café                      Park                    Pub                      Bakery
4    Central Toronto                                               Lawrence Park  43.728020 -79.388790        2.0
Park              Bus Line           Swim School       Department Store               Electronics Store
5    Central Toronto                                            Davisville North  43.712751 -79.390197        0.0
Park       Department Store                Hotel           Dance Studio               Food & Drink Shop
```

Thank you for your valuable time and careful browsing. Thanks to Coursera Team and Peers.

5. Discussion and Conclusion

This project is a good example of how to do data analysis and modeling. This project shows the whole process of data analysis from the initial data download or loading to data cleaning, to data analysis and final comparison. The only regret is that in the process of the project, I gradually realized that the initial assumption was unreasonable to a certain extent. In the process of the project, I constantly think about and gradually improve the direction and method of data analysis and modeling. In the end, the project gave me my own comparison goals and results.

There are also some possible future work on this project. For example, whether it is possible to show the final comparison results in the map. Or, if we take the lead in visual analysis on the map, is it possible to compare the visual analysis to get a higher level of conclusion?

I am very grateful to myself for this learning opportunity and to peers who have graded the quizzes and projects for me. Thanks to all the people who helped me in the learning process.