Abidzar Muhammad Ghifary Kurniawan
Coursera Capstone Project
For IBM Professional Certification in Data Science

Clustering Neighborhoods of New York, Toronto and San Francisco

---

Introduction

New York and San Francisco are cities that located in the USA, but does this fact make the two cities resemble each other in the matter of their similarity of famous venues type?

for this project, we will find similarity between two city with another city (Toronto) as a comparison of whether those two cities resemble each other or not according to their type of famous places retrieved from www.foursquare.com

---

Data Overview

Data: For the data of this project, I will refer to number of sources.

1. San Francisco neighborhoods:
    - I will be using Geojson file that provided by DataSF, and then scrape it to gain their San Francisco's neighborhood name, But this data lacks the coordinates too, I will obtain them using geopy library
    - https://data.sfgov.org/api/views/6ia5-2f8k/rows.json?accessType=DOWNLOAD

2. Toronto neighborhoods:
    - I will scrape the following Wikipedia page, using the BeautifulSoup package to obtain and transform the data in the table of postal codes on the page and transform it into a pandas dataframe. But this data lacks the coordinates too, I will obtain them from a link to a csv file that has the geographical coordinates of each postal code in Toronto. Later I will concatenate two dataset using methods provided in pandas library.
    - https://en.wikipedia.org/wiki/List_of_postal_codes_of_Canada:_M
    - http://cocl.us/Geospatial_data

3. New York neighborhoods:
    - I will convert addresses, retrieved from IBM server (so you can simply run a wget command and access the data), into their equivalent latitude and longitude values.
    - https://ibm.box.com/shared/static/fbpwbovar7lf8p5sgddm06cgipa2rxpe.json

Also, I will use the Foursquare API to explore neighborhoods in the cities. I will use the explore endpoint to get the most common venue categories in each neighborhood, and then use this feature to group the neighborhoods into clusters.

Abidzar Muhammad Ghifary Kurniawan
Coursera Capstone Project

I will use the k-means clustering algorithm to complete this task. Finally, you will use the Folium library to visualize the neighborhoods in given three cities and their emerging clusters.

Methodology

1. Instal and Import several library that we will use for data analysis

```
[1]: import numpy as np # library to handle data in a vectorized manner

     import pandas as pd # library for data analsysis
     pd.set_option('display.max_columns', None)
     pd.set_option('display.max_rows', None)

     import json # library to handle JSON files

     #!conda install -c conda-forge geopy --yes # uncomment this line if you haven't completed the Foursquare API lab
     from geopy.geocoders import Nominatim # convert an address into latitude and longitude values

     import requests # library to handle requests
     from pandas.io.json import json_normalize # tranform JSON file into a pandas dataframe

     # Matplotlib and associated plotting modules
     import matplotlib.cm as cm
     import matplotlib.colors as colors
     import matplotlib.pyplot as plt

     from bs4 import BeautifulSoup #library to scrape the websites

     # import k-means from clustering stage
     from sklearn.cluster import KMeans
     from scipy import spatial

     #!conda install -c conda-forge folium=0.5.0 --yes # uncomment this line if you haven't completed the Foursquare API lab
     import folium # map rendering library

     from geopy import geocoders  # if there isn't coordinate in json file (San Francisco dataset)
     gn = geocoders.GeoNames(username='abidzar')
     from geopy.geocoders import Nominatim
     from geopy.exc import GeocoderTimedOut

     print('Libraries imported.')

Libraries imported.
```

Library

2. Collect the data from provided source (look at dat a overview section)

The dataframe has 98 neighborhoods.

	Neighborhood	Latitude	Longitude	City	Borough
0	Seacliff	36.97412	-121.91579	San Francisco	San Francisco
1	Lake Street	39.04700	-122.91418	San Francisco	San Francisco
2	Presidio Terrace	37.78826	-122.46080	San Francisco	San Francisco
3	Inner Richmond	37.90187	-122.34108	San Francisco	San Francisco
4	Sutro Heights	37.77826	-122.51108	San Francisco	San Francisco

The dataframe has 103 neighborhoods.

	City	Borough	Neighborhood	Latitude	Longitude
0	Toronto	North York	Parkwoods	43.753259	-79.329656
1	Toronto	North York	Victoria Village	43.725882	-79.315572
2	Toronto	Downtown Toronto	Harbourfront, Regent Park	43.654260	-79.360636
3	Toronto	North York	Lawrence Heights, Lawrence Manor	43.718518	-79.464763
4	Toronto	Queen's Park	Queen's Park	43.662301	-79.389494

The dataframe has 306 neighborhoods.

	City	Borough	Neighborhood	Latitude	Longitude
0	New York	Bronx	Wakefield	40.894705	-73.847201
1	New York	Bronx	Co-op City	40.874294	-73.829939
2	New York	Bronx	Eastchester	40.887556	-73.827806
3	New York	Bronx	Fieldston	40.895437	-73.905643
4	New York	Bronx	Riverdale	40.890834	-73.912585

San Francisco, Toronto and New York's Dataset

Abidzar Muhammad Ghifary Kurniawan
Coursera Capstone Project
For IBM Professional Certification in Data Science

3. Combine all the databases we get from step 2 into one large dataset

```
The dataframe has 507 neighborhoods.
```

	Borough	City	Latitude	Longitude	Neighborhood
0	Bronx	New York	40.894705	-73.847201	Wakefield
1	Bronx	New York	40.874294	-73.829939	Co-op City
2	Bronx	New York	40.887556	-73.827806	Eastchester
3	Bronx	New York	40.895437	-73.905643	Fieldston
4	Bronx	New York	40.890834	-73.912585	Riverdale

4. Using large dataset, find the nearby venues on the each neighborhood on each city using Foursquare API

Define Foursquare Credentials and Version

In [29]:
```
CLIENT_ID =                                                  # your Foursquare ID
CLIENT_SECRET =                                              # your Foursquare Secret
VERSION = '20180605' # Foursquare API version
```

define a function to get the nearby venues of Neighborhoods

In [30]:
```
LIMIT = 50

def getNearbyVenues(cities, names, latitudes, longitudes, radius=500):
    i=0
    venues_list=[]
    for city, name, lat, lng in zip(cities, names, latitudes, longitudes):
        i+=1
        print(i, city, name)

        # create the API request URL
        url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&r
adius={}&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([(
            city,
            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 = ['City',
                  'Neighborhood',
                  'Neighborhood Latitude',
                  'Neighborhood Longitude',
                  'Venue',
                  'Venue Latitude',
                  'Venue Longitude',
                  'Venue Category']

    return(nearby_venues)
```

Process that we use for this step

```
all_venues = getNearbyVenues(cities = merged_neighborhood['City'], names = merged_neighborhood['Neighborhood'],
                             latitudes = merged_neighborhood['Latitude'],
                             longitudes = merged_neighborhood['Longitude'])
```

```
print(all_venues.shape)
all_venues.head()
```

```
(20996, 7)
```

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude	Venue Category
0	Wakefield	40.894705	-73.847201	Lollipops Gelato	40.894123	-73.845892	Dessert Shop
1	Wakefield	40.894705	-73.847201	Rite Aid	40.896521	-73.844680	Pharmacy
2	Wakefield	40.894705	-73.847201	Carvel Ice Cream	40.890487	-73.848568	Ice Cream Shop
3	Wakefield	40.894705	-73.847201	Dunkin Donuts	40.890631	-73.849027	Donut Shop
4	Wakefield	40.894705	-73.847201	SUBWAY	40.890656	-73.849192	Sandwich Place

output dataset

5. Find the top 10 venues for each neighborhood

First, let's write a function to sort the venues in descending order.

```
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]
```

Now let's create the new dataframe and display the top 10 venues for each neighborhood.

```
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'] = venues_grouped['Neighborhood']

for ind in np.arange(venues_grouped.shape[0]):
    neighborhoods_venues_sorted.iloc[ind, 1:] = return_most_common_venues(venues_grouped.iloc[ind, :], num
_top_venues)
```

Process that we use for this step

```
neighborhoods_venues_sorted.head()
```

	Neighborhood	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
0	Adelaide, King, Richmond	Steakhouse	Café	American Restaurant	Coffee Shop	Breakfast Spot	Restaurant	Hotel	Bar	Asian Restaurar
1	Agincourt	Skating Rink	Breakfast Spot	Lounge	Clothing Store	Yoga Studio	Farmers Market	English Restaurant	Ethiopian Restaurant	Event Service
2	Agincourt North, L'Amoreaux East, Milliken, St...	Park	Playground	Yoga Studio	Farmers Market	Empanada Restaurant	English Restaurant	Ethiopian Restaurant	Event Service	Event Space
3	Alamo Square	Bar	Sushi Restaurant	Wine Bar	Pizza Place	Seafood Restaurant	Mediterranean Restaurant	Cocktail Bar	Rock Club	BBQ Joint
4	Albion Gardens, Beaumond Heights, Humbergate, ...	Grocery Store	Coffee Shop	Fried Chicken Joint	Pharmacy	Video Store	Fast Food Restaurant	Sandwich Place	Pizza Place	Beer Store

Output dataset

6. Cluster the neighborhood using k-means clustering into 5 clusters based on their top 10 venues

```
# set number of clusters
kclusters = 5

venues_clustering = venues_grouped.drop('Neighborhood', 1)

# run k-means clustering
kmeans = KMeans(n_clusters=kclusters, random_state=42).fit(venues_clustering)

# check cluster labels generated for each row in the dataframe
kmeans.labels_[0:10]
```

```
array([3, 3, 4, 3, 0, 0, 0, 3, 0, 2], dtype=int32)
```

```
venues_merged = neighborhoods_venues_sorted

# add clustering labels
venues_merged['Cluster Labels'] = kmeans.labels_

# merge df_grouped with city data to add latitude/longitude for each neighborhood
venues_merged = venues_merged.join(all.set_index('Neighborhood'), on='Neighborhood')

venues_merged.head() # check the last columns!
```

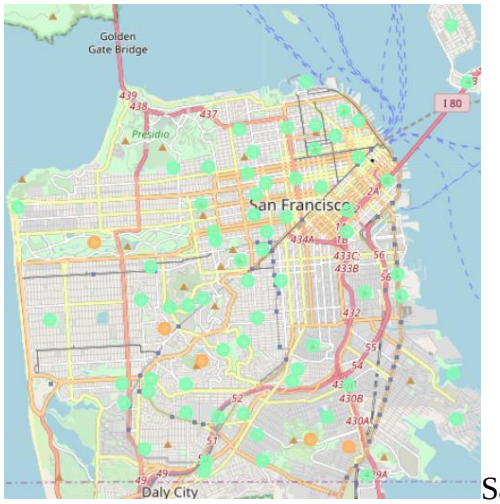
Process that we use for this step

Abidzar Muhammad Ghifary Kurniawan
Coursera Capstone Project
For IBM Professional Certification in Data Science

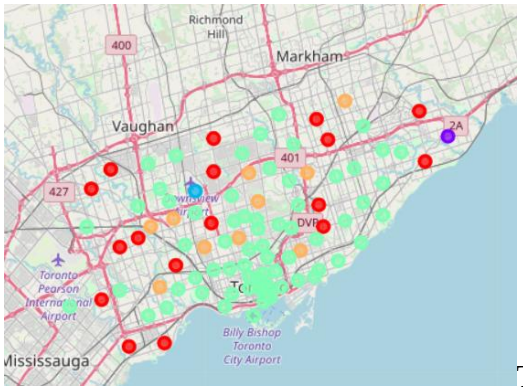
	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue	9th Most Common Venue
0	Adelaide, King, Richmond	Steakhouse	Café	American Restaurant	Coffee Shop	Breakfast Spot	Restaurant	Hotel	Bar	Asian Restaura
1	Agincourt	Skating Rink	Breakfast Spot	Lounge	Clothing Store	Yoga Studio	Farmers Market	English Restaurant	Ethiopian Restaurant	Event Service
2	Agincourt North, L'Amoreaux East, Milliken, St...	Park	Playground	Yoga Studio	Farmers Market	Empanada Restaurant	English Restaurant	Ethiopian Restaurant	Event Service	Event Space
3	Alamo Square	Bar	Sushi Restaurant	Wine Bar	Pizza Place	Seafood Restaurant	Mediterranean Restaurant	Cocktail Bar	Rock Club	BBQ Join
4	Albion Gardens, Beaumond Heights, Humbergate, ...	Grocery Store	Coffee Shop	Fried Chicken Joint	Pharmacy	Video Store	Fast Food Restaurant	Sandwich Place	Pizza Place	Beer Stor

Output dataset

7. Visualize into the map

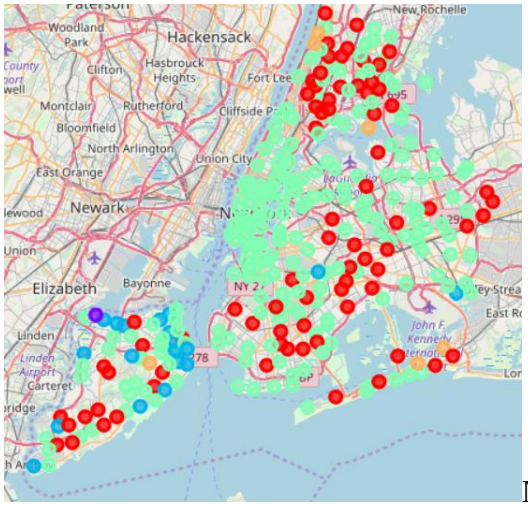


San Francisco



Toronto

Abidzar Muhammad Ghifary Kurniawan
Coursera Capstone Project
For IBM Professional Certification in Data Science



New York

8. find the similarities between cities by implementing Cosine similarity test
    - find the number of every cluster on every city

```
clusters_total = venues_merged[['City','Cluster Labels']].groupby('City',sort=False).count().rename(columns={'Cluster Labels':'Total'}).transpose()
clusters_total.columns.name = None
clusters_total

df_desc = pd.concat([venues_merged[['City','Cluster Labels']].groupby('City').get_group(city).groupby('Cluster Labels').count().rename(
        columns={'City':city}) for city in venues_merged['City'].unique().tolist()], axis=1, sort=False)
df_desc = df_desc.append(clusters_total).astype(float)
df_desc
```

	Toronto	San Francisco	New York
0	17.0	3.0	76.0
1	1.0	1.0	1.0
2	1.0	1.0	19.0
3	71.0	81.0	204.0
4	10.0	8.0	6.0
Total	100.0	94.0	306.0

    - and then convert it to percentage

```
#for easier task, convert the dataframe into percentage
df_perc = df_desc/df_desc.iloc[5]
df_perc
```

	Toronto	San Francisco	New York
0	0.17	0.031915	0.248366
1	0.01	0.010638	0.003268
2	0.01	0.010638	0.062092
3	0.71	0.861702	0.666667
4	0.10	0.085106	0.019608
Total	1.00	1.000000	1.000000

    - find the similarities

Abidzar Muhammad Ghifary Kurniawan
Coursera Capstone Project
For IBM Professional Certification in Data Science

```
list_sim = []
list_cities = []

for city1 in df_perc.columns:
    for city2 in df_perc.columns[df_perc.columns.tolist().index(city1)+1:]:
        result = 1 - spatial.distance.cosine(df_perc.iloc[:5][city1], df_perc.iloc[:5][city2])
        list_sim.append(result)
        list_cities.append(city1+' & '+city2)

df_similarity = pd.DataFrame({'Similarity':list_sim}, index=list_cities)
df_similarity
```

	Similarity
Toronto & San Francisco	0.980031
Toronto & New York	0.984114
San Francisco & New York	0.944505

Report:

I am using two machine learning technique :

1. **K-means clustering**, to grouping all neighborhood from San Francisco, Toronto, and New York into five cluster
2. **cosine similarity,** to find similiarity between three cities in percentage

	Similarity
Toronto & San Francisco	0.980031
Toronto & New York	0.984114
San Francisco & New York	0.944505

As illustrated on the table above, **Toronto and New York** cities have the highest Cosine similarity of 98.4 **%**, while the least similar cities among three are **San Francisco and New York** with Cosine similarity of **94.4 %**.

Discussion:

Another point that is noteworthy is the level of similarity between San Francisco and Toronto which reached 98% and New York and Toronto which reached 98.4%, even though Toronto is in a different country with these two cities

---

Abidzar Muhammad Ghifary Kurniawan
Coursera Capstone Project
For IBM Professional Certification in Data Science

Conclusion:

As my final conclusion, for this project, I want to proving the hypothesis that San Francisco are resemblances of New York based on places that are quite popular in their neighborhood

the way I find their neighborhood with either data scrapping, download json file or download csv file that contain their neighborhood name and then I find the nearest venues from their neighborhood using Foursquare API after that, I am clustering them using a machine learning technique K-means clustering algorithm into five cluster, finally, the outcome is subjected to Cosine similarity test to compare the similarities between the cities.

what we can get from the project is that comparing to the similarity level between San Francisco with Toronto and New York with Toronto, the similarity level San Francisco and New York subjectively small, which mean my hypothesis is not proven.