Capstone Project

The Battle of Neighborhoods

（Full Report）

Zhang Yuan

Index

Introduction

Problem Statement

Toronto is the most populous city in Canada. It is recognized as one of the most multicultural and cosmopolitan cities in the world, and there are lots of different style restaurants in Toronto, making it difficult to make a choice for visitors or even for local citizens. Data science can help to solve this problem using recommendation systems. Recommendation systems are a collection of algorithms used to recommend items to users based on information taken from the user. So the problem we try to solve here is：
how to recommend restaurants to a specific user based on his own preferences?

Objectives:

This project will use recommendation algorithms to implement a simple version of one, which try to recommend restaurants based on user's location and input of restaurants they've been and their ratings.

Target Audience:

Through this project we are expecting following people to benefit out of the findings.

- Tourist.
- People migrating city for work

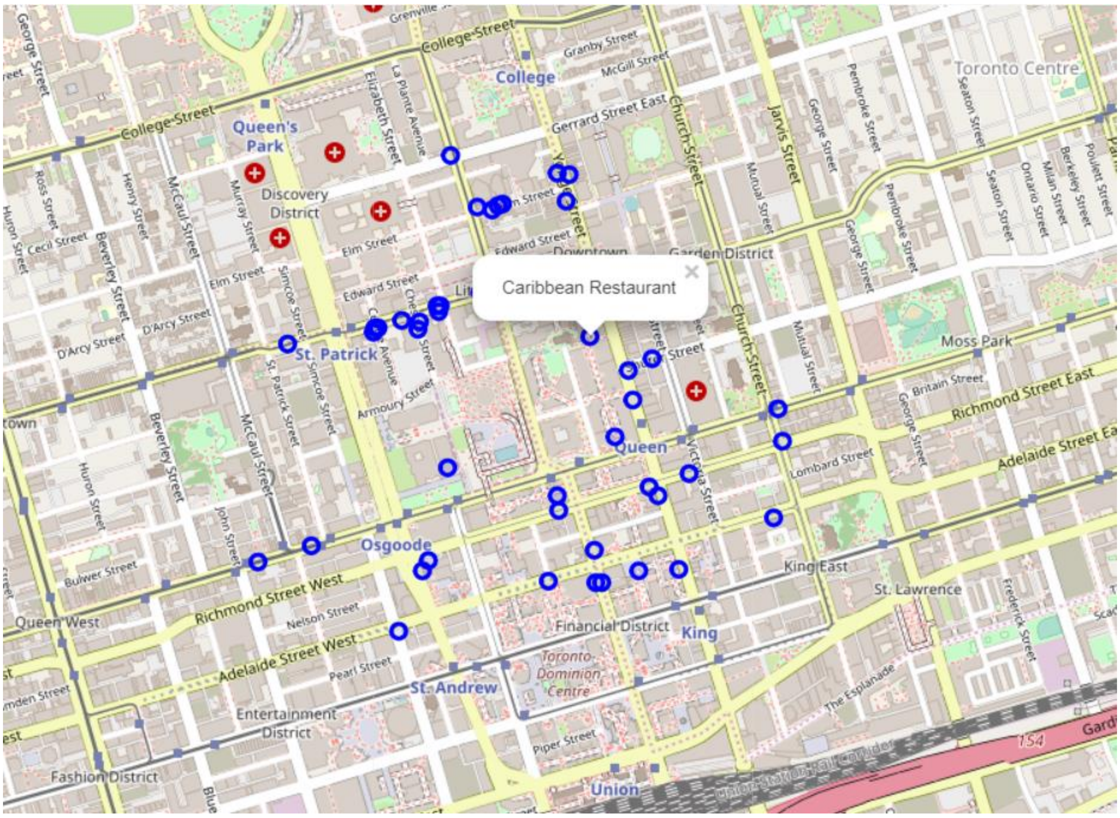
Data

Data we'll use in this project is from foursquare::

1) First we'll use "**Search for Venues**" API to get a list of restaurants near the current location. Response JSON data includes:

Field	**Description**
id	A unique string identifier for this venue.
name	The best known name for this venue.
location	An object containing none, some, or all of address (street address), crossStreet, city, state, postalCode, country, lat, lng, and distance. All fields are strings, except for lat, lng, and distance. Distance is measured in meters. Some venues have their locations intentionally hidden for privacy reasons (such as private residences). If this is the case, the parameter isFuzzed will be set to true, and the lat/lng parameters will have reduced precision.
categories	An array, possibly empty, of categories that have been applied to this venue. One of the categories will have a primary field indicating that it is the primary category for the venue. For the complete category tree, see categories.

After we get the list, we can use Folium to draw a map：



2) Then we'll use "**Get Details of a Venue**" API to get a detail information of each venue near the current location and store into a DataFrame. There are a lot of information in the response JSON data, but we'll only keep few fields which are useful for the project:

Field	Description
id	A unique string identifier for this venue.
name	The best known name for this venue.
location	An object containing none, some, or all of address (street address), crossStreet, city, state, postalCode, country, lat, lng, and distance. All fields are strings, except for lat, lng, and distance. Distance is measured in meters. Some venues have their locations intentionally hidden for privacy reasons (such as private residences). If this is the case, the parameter is Fuzzed will be set to true, and the lat/lng parameters will have reduced precision.
categories	An array, possibly empty, of categories that have been applied to this venue. One of the categories will have a primary field indicating that it is the primary category for the venue. For the complete category tree, see categories.
stats	Contains checkinsCount (total checkins ever here), usersCount (total users who have ever checked in here), and tipCount (number of tips here).
price	An object containing the price tier from 1 (least pricey) - 4 (most pricey) and a message describing the price tier.
rating	Numerical rating of the venue (0 through 10). Returned as part of an explore result, excluded in search results. Not all venues will have a rating.

description	Description of the venue provided by venue owner.
tips	Contains the total count of tips and groups with friends and others as groupTypes. Groups may change over time.
likes	The count of users who have liked this venue, and groups containing any friends and others who have liked it. The groups included are subject to change.
attributes	Attributes associated with the venue, such as price tier, whether the venue takes reservations, and parking availability.

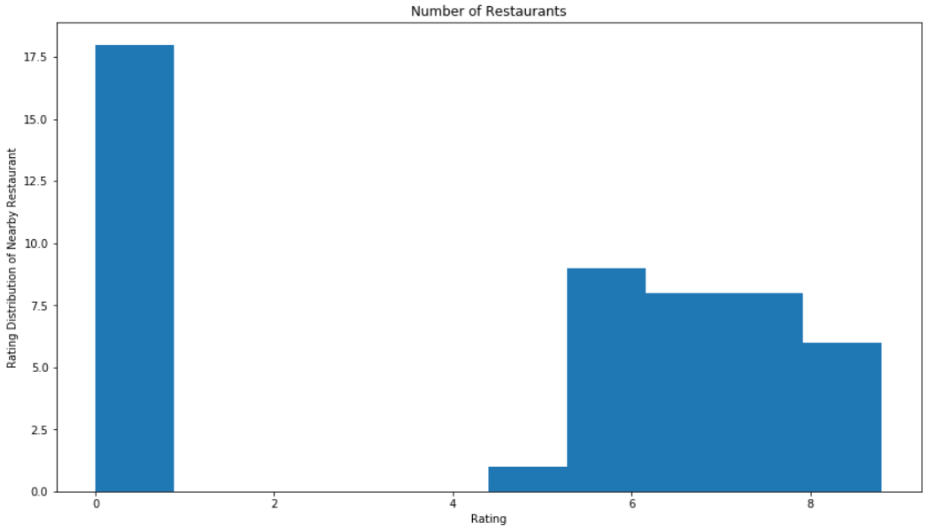
The result dataframe is like following one, which displays the above info ordered by ratings:

Out[33]:

	name	categories	lat	lng	distance	Rating	Pricing	Likes	tips
0	Terroni	Italian Restaurant	43.650927	-79.375602	664	8.8	3.0	267.0	94.0
1	Salad King	Thai Restaurant	43.657601	-79.381620	572	8.6	2.0	490.0	209.0
2	The Elm Tree Restaurant	Modern European Restaurant	43.657397	-79.383761	529	8.3	0.0	23.0	12.0
3	The Senator Restaurant	Diner	43.655641	-79.379199	481	8.2	2.0	183.0	96.0
4	JOEY	American Restaurant	43.656094	-79.381878	405	8.1	2.0	348.0	179.0
5	Little India Restaurant	Indian Restaurant	43.650319	-79.388998	512	8.0	2.0	127.0	75.0
6	Reds Wine Tavern	Gastropub	43.649570	-79.382129	359	7.8	3.0	94.0	35.0
7	Fune Japanese Restaurant	Japanese Restaurant	43.648514	-79.386457	517	7.7	2.0	52.0	27.0
8	Ali Baba's	Middle Eastern Restaurant	43.654916	-79.387172	388	7.5	1.0	18.0	3.0
9	Yueh Tung Chinese Restaurant	Chinese Restaurant	43.655281	-79.385337	328	7.5	1.0	29.0	10.0
10	Mercatto	Italian Restaurant	43.650243	-79.380820	344	7.4	3.0	45.0	26.0
11	Lai Wah Heen	Chinese Restaurant	43.655038	-79.385890	328	7.4	3.0	38.0	50.0
12	Hong Shing Chinese Restaurant	Chinese Restaurant	43.654925	-79.387089	384	7.3	2.0	35.0	32.0
13	Fran's	Diner	43.654265	-79.379120	396	7.1	2.0	131.0	64.0
15	Golden Thai Restaurant	Thai Restaurant	43.652525	-79.375369	655	7.0	3.0	40.0	31.0
16	Donatello Restaurant	Italian Restaurant	43.657489	-79.383605	539	7.0	3.0	20.0	19.0
14	Adega Restaurant	Restaurant	43.657519	-79.383462	542	7.0	3.0	15.0	13.0
17	Tundra Restaurant	Restaurant	43.650010	-79.385608	338	6.8	4.0	24.0	12.0
18	McDonald's	Fast Food Restaurant	43.658196	-79.381872	631	6.4	1.0	56.0	17.0
19	Hemispheres Restaurant & Bistro	American Restaurant	43.654884	-79.385931	316	6.3	1.0	9.0	5.0
20	Hendricks Restaurant & Bar	Restaurant	43.653415	-79.379698	318	6.3	2.0	10.0	5.0
21	Richtree Natural Market Restaurants	Restaurant	43.652614	-79.380231	263	6.2	2.0	109.0	49.0
22	Akashiro Japanese Restaurant & Bar	Sushi Restaurant	43.655965	-79.380541	440	5.9	2.0	18.0	10.0
23	Wah Too Seafood Restaurant	Chinese Restaurant	43.654833	-79.387206	384	5.9	1.0	4.0	11.0
24	Kyoto House Japanese Restaurant	Sushi Restaurant	43.655381	-79.385270	336	5.8	2.0	72.0	55.0
27	McDonald's	Fast Food Restaurant	43.653215	-79.375487	649	5.7	1.0	17.0	9.0
25	Ninki Sushi	Japanese Restaurant	43.649812	-79.379518	450	5.7	3.0	5.0	15.0

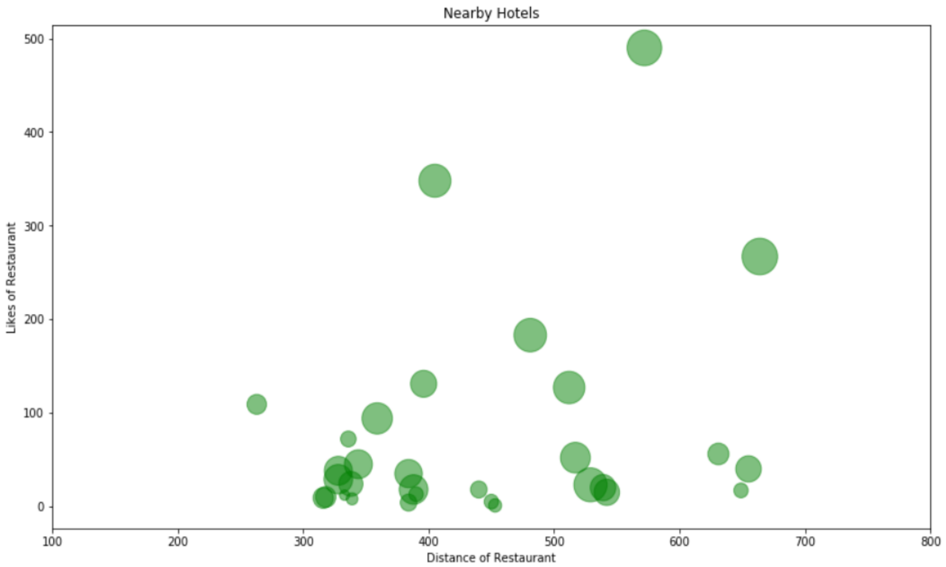
3) Let's make some visualization about data.

First let's view the rating distribution of nearby restaurants:



Then let's view the rating (bubble size) and relevant distance and number of likes.

```
Out[47]: Text(0.5, 1, 'Nearby Hotels')
```



We can further explore the users who like the restaurant (for example, restaurant JOEY):

```
#Get Users who like the venue
venue_id = '4df909dfe4cd2129701c0690' # ID of JOEY
url = 'https://api.foursquare.com/v2/venues/{}/likes?client_id={}&client_secret={}&v={}'.format(venue_id, CLIENT_ID, CLIENT_SECRET, VERSION)
url

results = requests.get(url).json()
user_results = json_normalize(results['response']['likes']['items'])
```

```
user_results
```

Out[27]:

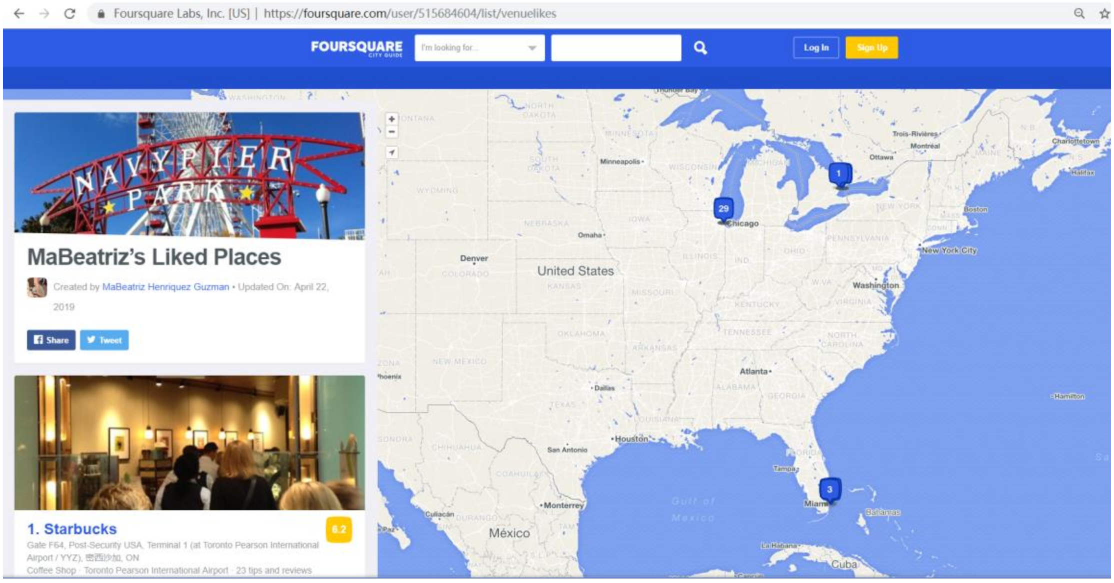
	firstName	gender	id	lastName	photo.prefix	photo.suffix
0	MaBeatriz	female	515684604	Henriquez Guzman	https://fastly.4sqi.net/img/user/	/515684604_Asaxe-WJ_oGWaooY0VwNq0I31RjnDrCy2Zp...
1	Adeline	female	480391294	Garcia	https://fastly.4sqi.net/img/user/	/480391294_5LID1iJx_YBKzDowE-_4-IlpLoqfZKuvG9i...
2	Pedro	male	60244990	Vaccara	https://fastly.4sqi.net/img/user/	/TZYRTJA4F05G0LTS.jpg

Then pull image of some user (MaBeatriz)

In [58]:
```
Image('https://fastly.4sqi.net/img/user/300x300/515684604_Asaxe-WJ_oGWaooYOVwNqOI31RjnDrCy2ZpV_9EOqEi3vZiF13D0j-DVGj8N145e-UqwsmFhI.jpg')
```

Out[58]:

And the venues the user likes:



However, it seems more privileges are required to call API to get venues the user likes.

In [62]:
```
user_id = '515684604' # user ID with most agree counts and complete profile

url = 'https://api.foursquare.com/v2/users/{}/venuelikes?client_id={}&client_secret={}&v={}'.format(user_id, CLIENT_ID, CLIENT_SECRET, VERSION) # define URL

# send GET request
results = requests.get(url).json()
results
```

Out[62]:
```
{'meta': {'code': 403,
  'errorDetail': 'A user is required to call this endpoint.',
  'errorType': 'not_authorized',
  'requestId': '5ccbeeccdb04f559d9b0a285'},
 'response': {}}
```

Methodology

After explored the data, now let's build a simple recommendation system.

Generally, there are 2 main types of machine learning recommendation systems: content-based and collaborative filtering (user-based). Content-based systems try to figure out what a user's favorite aspects of an item are, and then make recommendations on items that share those aspects. Collaborative filtering techniques find similar groups of users, and provide recommendations based on similar tastes within that group.

Typically, user-based collaborative filtering is more frequently used to recommend restaurants because it's more like a social event. But due to limitation of data we can acquired from Foursquare, let's try to use **item-based recommendation** instead.

Now let's suppose the user's name is Jack. To do recommendation, first we need to get input data of restaurants Jack has been and his own rating:

In [14]:
```
userInput = [
            {'name':'Terroni', 'user_Rating':1},
            {'name':'Cali Restaurant', 'user_Rating':8}
         ]
inputRestaurant = pd.DataFrame(userInput)
inputRestaurant
```

Out[14]:

	name	user_Rating
0	Terroni	1
1	Cali Restaurant	8

Then filtering out the restaurants from the input and get the quantitive features of user restaurants, then compute the weights of Jack when he choose restaurant:

Filtering out the restaurants from the input

In [16]:
```
userRestaurant = dataframe_filtered[dataframe_filtered['id'].isin(inputRestaurant['id'].tolist())]
userRestaurant
```

Out[16]:

	id	name	categories	address	cc	city	country	crossStreet	distance	formattedAddress	labeledLatLngs	lat	lng
41	4b49183ff964a520a46526e3	Terroni	Italian Restaurant	57 Adelaide St. E	CA	Toronto	Canada	at Church St.	0.005988	['57 Adelaide St. E (at Church St.)', 'Toronto...	[{'lng': -79.375602, 'label': 'display', 'lat'...	43.650927	-79.375602
20	4c476d6719fde21e32410876	Cali Restaurant	Vietnamese Restaurant	179 Dundas St. W.	CA	Toronto	Canada	at Chestnut	0.468563	['179 Dundas St. W. (at Chestnut)', 'Toronto O...	[{'lng': -79.386375, 'label': 'display', 'lat'...	43.655068	-79.386375

Get the quantitive features of user restaurants

In [17]:
```
#Resetting the index to avoid future issues
userRestaurant = userRestaurant.reset_index(drop=True)
#Dropping unnecessary issues due to save memory and to avoid issues
userRestaurantTable = userRestaurant[['distance','Rating','Pricing','Likes','tips']]

userRestaurantTable
```

Out[17]:

	distance	Rating	Pricing	Likes	tips
0	0.005988	1.0	0.25	0.544898	0.449761
1	0.468563	0.0	0.50	0.000000	0.000000

Compute Jack's weights based on his input ,you can see Jack put more weights on distance and price

[20]:
```
#userGenreTable.transpose()
#Dot produt to get weights
userProfile = userRestaurantTable.transpose().dot(inputRestaurant['user_Rating'])
#The user profile
userProfile
```

Out[20]:
```
distance    3.754491
Rating      1.000000
Pricing     4.250000
Likes       0.544898
tips        0.449761
dtype: float64
```

Now let's get the features of every restaurant in our original dataframe

In [21]:

```python
restaurantTable = dataframe_filtered.set_index(dataframe_filtered['id'])
#And drop the unnecessary information
restaurantTable = restaurantTable[['distance','Rating','Pricing','Likes','tips']]
restaurantTable.head()
```

Out[21]:

id	distance	Rating	Pricing	Likes	tips
4b49183ff964a520a46526e3	0.005988	1.000000	0.25	0.544898	0.449761
4ad4c061f964a52095f720e3	0.143713	0.977273	0.50	1.000000	1.000000
539c6f13498e06f4cc765165	0.208084	0.943182	1.00	0.046939	0.057416
4ad7929cf964a520500c21e3	0.279940	0.931818	0.50	0.373469	0.459330
4df909dfe4cd2129701c0690	0.393713	0.920455	0.50	0.710204	0.856459

In [22]:

```python
restaurantTable.shape
```

Out[22]: (50, 5)

Multiply the features by the weights and then take the weighted average

In [23]:

```python
recommendationTable_df = ((restaurantTable*userProfile).sum(axis=1))/(userProfile.sum())
recommendationTable_df.head()
```

Out[23]:

```
id
4b49183ff964a520a46526e3    0.258440
4ad4c061f964a52095f720e3    0.463689
539c6f13498e06f4cc765165    0.602634
4ad7929cf964a520500c21e3    0.451833
```

Sort our recommendations in descending order

[24]:

```python
recommendationTable_df = recommendationTable_df.sort_values(ascending=False)
#Just a peek at the values
recommendationTable_df.head()
```

Out[24]:

```
id
539c6f13498e06f4cc765165    0.602634
52a7ae41498eed3af4d0a3fa    0.600502
4ad4c05ff964a52048f720e3    0.590309
56dd9d68498eb4e5edcb30f9    0.584110
4ddd83c788779c82beb061fc    0.564046
dtype: float64
```

Results

After sort all restaurant's features based on weights of Jack, we get the recommendation. You can see the recommendation order is different from foursquare's rating order, that's because Jack has his own weights (price and distance)

	user_rating_score	name	categories	lat	lng	distance	Rating	Pricing	Likes	tips
id										
539c6f13498e06f4cc765165	0.602634	The Elm Tree Restaurant	Modern European Restaurant	43.657397	-79.383761	0.208084	0.943182	1.00	0.046939	0.057416
52a7ae41498eed3af4d0a3fa	0.600502	Yueh Tung Chinese Restaurant	Chinese Restaurant	43.655281	-79.385337	0.508982	0.852273	0.75	0.059184	0.047847
4ad4c05ff964a52048f720e3	0.590309	Hemispheres Restaurant & Bistro	American Restaurant	43.654884	-79.385931	0.526946	0.715909	0.75	0.018367	0.023923
56dd9d68498eb4e5edcb30f9	0.584110	Spring Rolls	Japanese Restaurant in Toronto	Theme Restaurant	43.656105	-79.383495	0.423653	0.000000	1.00	0.000000	0.000000
4ddd83c788779c82beb061fc	0.564046	Ali Baba's	Middle Eastern Restaurant	43.654916	-79.387172	0.419162	0.852273	0.75	0.036735	0.014354

Discussion

Advantages:

- Learns user's preferences
- Highly personalized for the user

Disadvantages:

- Extracting data is not always intuitive
- Determining what characteristics of the item the user dislikes or likes is not always obvious

Limitation:

- Data and access limitation of foursquare free account

Conclusion

Foursquare provides rich location information about venues, users and more. Use the data and item-based machine learning algorithm together, it's possible for us to build a recommend system to recommend restaurants to users based on his own preferences. It's also possible to recommend other categories venues to users in similar way.