DATA_301_Assignment_5B_ISHAAN_SATHAYE_SRESHTA_TALLURI

November 6, 2023

1 Bing Maps API

In this part of the lab, you will join the CityBikes data from the previous part of the lab with additional data that you will query from the Bing Maps API.

First, you will need to register for a Bing Maps Key. Follow the instructions here. You should be able to sign in with your Cal Poly account. When you get to Step 4, select:

- Key type: Basic
- Application type: Dev/Test

You will be able to make 125000 free requests with the resulting API key. This should be more than enough to complete this assignment.

We will be working with the REST services in the Bing Maps API. Click on the link for a complete documentation of the features.

[52]:
```
import pandas as pd
import numpy as np
import requests
import time
import json
```

1.1 Question 1

Read in the `DataFrame` of bike stations in the United States from Part A of this lab. Restrict to the stations in the "Bay Wheels" network (with network ID "ford-gobike").

How many of these stations are in the city/county of San Francisco?

(*Hint:* Use the Locations API to get the address associated with each latitude and longitude coordinate.)

[17]:
```
df_bike_stations = pd.read_csv('./bikes_us.csv')
df_bike_stations.head()
```

[17]:
```
   Unnamed: 0                                            company
0          28                ['PBSC', 'Alta Bicycle Share, Inc']  \
1          81                                   ['BCycle, LLC']
2          82            ['Motivate International, Inc', 'PBSC']
```

```
3          85  ['Portland Bureau of Transportation (PBOT)', '…
4          87                                  ['BCycle, LLC']

                           href                id              name
0          /v2/networks/we-cycle          we-cycle          WE-cycle  \
1            /v2/networks/austin            austin    Austin B-cycle
2  /v2/networks/bike-chattanooga  bike-chattanooga  Bike Chattanooga
3          /v2/networks/biketown          biketown          BIKETOWN
4           /v2/networks/boulder           boulder   Boulder B-cycle

     location.city location.country  location.latitude  location.longitude
0        Aspen, CO               US          39.194951         -106.837002  \
1       Austin, TX               US          30.264080          -97.743550
2  Chattanooga, TN               US          35.045630          -85.309680
3     Portland, OR               US          45.521754         -122.681079
4      Boulder, CO               US          40.008110         -105.263850

   source                                          gbfs_href  license.name
0     NaN  https://asp.publicbikesystem.net/ube/gbfs/v1/g…           NaN  \
1     NaN    https://gbfs.bcycle.com/bcycle_austin/gbfs.json           NaN
2     NaN     https://chat.publicbikesystem.net/ube/gbfs/v1/           NaN
3     NaN        https://gbfs.biketownpdx.com/gbfs/gbfs.json           NaN
4     NaN   https://gbfs.bcycle.com/bcycle_boulder/gbfs.json           NaN

   license.url ebikes
0          NaN    NaN
1          NaN    NaN
2          NaN   True
3          NaN    NaN
4          NaN    NaN
```

[18]:
```
df_bay_wheels = df_bike_stations[df_bike_stations['id'] == 'bay-wheels']
df_bay_wheels.head()
```

[18]:
```
    Unnamed: 0          company                      href          id
31         409  ['Motivate LLC']  /v2/networks/bay-wheels  bay-wheels  \

          name              location.city location.country
31  Bay Wheels  San Francisco Bay Area, CA               US  \

    location.latitude  location.longitude  source
31          37.714145             -122.25     NaN  \

                                  gbfs_href  license.name  license.url
31  https://gbfs.baywheels.com/gbfs/gbfs.json           NaN          NaN  \

   ebikes
```

```
31    NaN
```

[24]:
```
url = "http://api.citybik.es" + df_bay_wheels['href'][31]
resp = requests.get(url)
df_stations = pd.json_normalize(resp.json().get('network').get('stations'))
df_stations.head()
```

[24]:

	empty_slots	free_bikes	id	latitude	longitude	name	timestamp	extra.ebikes	extra.has_ebikes	extra.last_updated	extra.payment	extra.payment-terminal	extra.rental_uris.android	extra.rental_uris.ios	extra.renting	extra.returning	extra.slots	extra.uid	extra.address
0	8	7	d0e8f4f1834b7b33a3faf8882f567ab8	37.849735	-122.270582	Harmon St at Adeline St	2023-11-07T06:36:44.353000Z	0	True	1699338936	[key, creditcard]	True	https://sfo.lft.to/lastmile_qr_scan	https://sfo.lft.to/lastmile_qr_scan	1	1	15	fd89514c-f878-4cd5-8113-8e5beead44de	NaN
1	1	10	983514094dd808b1604da2dcfc2d09af	37.336188	-121.889277	Fountain Alley at S 2nd St	2023-11-07T06:36:43.430000Z	0	True	1699338946	[key, creditcard]	True	https://sfo.lft.to/lastmile_qr_scan	https://sfo.lft.to/lastmile_qr_scan	1	1	11	d12ba265-5bfe-4a00-a19a-a7299822bd65	NaN
2	16	7	da17603652106fda93da4e255a5b0a22	37.322125	-121.881090	Oak St at 1st St	2023-11-07T06:36:44.306000Z	0	True	1699338982	[key, creditcard]	True	https://sfo.lft.to/lastmile_qr_scan	https://sfo.lft.to/lastmile_qr_scan	1	1	23	0fd9a51c-67ac-4048-b531-bff644b82f47	NaN
3	6	8	7a21c92b3b4cd2f7759107b4fdebf869	37.323678	-121.874119	Bestor Art Park	2023-11-07T06:36:43.771000Z	1	True	1699338982	[key, creditcard]	True	https://sfo.lft.to/lastmile_qr_scan	https://sfo.lft.to/lastmile_qr_scan	1	1	15	46b4ef45-b06b-40eb-9fdf-9bc8ff104a4f	NaN
4	17	9	ce34d38fb230a23c1ced12d1e16df294	37.325998	-121.877120	5th St at Virginia St	2023-11-07T06:36:44.166000Z	1	True	1699338969	[key, creditcard]	True	https://sfo.lft.to/lastmile_qr_scan	https://sfo.lft.to/lastmile_qr_scan	1	1			

```
4           27  bb54f5c8-5578-4167-9599-8e55154de4fd          NaN
```

[38]:
```
len(df_stations)
```

[38]:
```
548
```

[42]:
```
sf_count = 0
df_stations_sf = pd.DataFrame(columns=['Station Name', 'Latitude', 'Longitude',␣
↪'Address'])
for idx, station in df_stations.iterrows():
  point = f'{station["latitude"]},{station["longitude"]}'
  auth = "Al3X4501N_LoywhAMfdbiDJe6Kd4cyPzon2xm55COHrQALhIX7Ay7qzBeSCa664r"
  response = requests.get(f"http://dev.virtualearth.net/REST/v1/Locations/
↪{point}?&key={auth}")
  address = response.
↪json()['resourceSets'][0]['resources'][0]['address']['formattedAddress']
  adminDistrict2 = response.
↪json()['resourceSets'][0]['resources'][0]['address']['adminDistrict2']
  if adminDistrict2 == 'San Francisco Co.':
    sf_count += 1
    df_stations_sf.loc[sf_count] = [station['name'], station['latitude'],␣
↪station['longitude'], address]

  time.sleep(0.5)
```

[43]:
```
df_stations_sf.head()
```

[43]:
```
                   Station Name   Latitude   Longitude
1          17th St at Dolores St  37.763015 -122.426497  \
2              4th St at 16th St  37.767045 -122.390833
3            Folsom St at 3rd St  37.783830 -122.398870
4  Gennessee St at Monterey Blvd  37.731657 -122.451122
5  The Embarcadero at Sansome St  37.804770 -122.403234

                                          Address
1  392-398 Dolores St, San Francisco, CA 94114, U…
2  1783 4th St, San Francisco, CA 94158, United S…
3  694 Folsom St, San Francisco, CA 94107, United…
4  696-698 Monterey Blvd, San Francisco, CA 94127…
5  1155 Sansome St, San Francisco, CA 94111, Unit…
```

[44]:
```
len(df_stations_sf)
```

[44]:
```
334
```

There are 334 Bay Wheel stations in San Francisco.

1.2 Question 2

You want to go to Coit Tower. To save money, you decide to ride a "Bay Wheels" bike to the closest station and hail a cab from there to Coit Tower. What station should you bike to so that you are as close to Coit Tower as possible (as measured by driving distance)? Does your answer agree with the one that you obtained in Part A of this lab? If not, why does it differ?

Hints: - You should restrict your attention to bike stations that are in San Francisco, which you determined in Question 1. - Use the Routes API to calculate a distance matrix between Coit Tower and the bike stations. - You can do this with just one call to the API. Because there are too many stations, it is impossible to specify all the locations in the URL. Instead, you should make a POST request (`requests.post`), passing in the parameters as a JSON object through the `json=` parameter of `requests.post`. Read the API documentation carefully to learn how to use the POST API.

[86]:
```
origins = df_stations_sf[['Latitude', 'Longitude']].to_dict('records')

body = {
    "origins": origins,
    "destinations": [{'latitude': 37.802139, 'longitude': -122.405853}],
    "travelMode": "driving",
}

url = f"https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrixAsync?
 ↪key={auth}"

response = requests.post(url, json=body)
response.json()
```

[86]:
```
{'authenticationResultCode': 'ValidCredentials',
 'brandLogoUri': 'https://dev.virtualearth.net/Branding/logo_powered_by.png',
 'copyright': 'Copyright © 2023 Microsoft and its suppliers. All rights
reserved. This API cannot be accessed and the content and any results may not be
used, reproduced or transmitted in any manner without express written permission
from Microsoft Corporation.',
 'resourceSets': [{'estimatedTotal': 1,
   'resources': [{'__type':
'RouteProxyAsyncResult:http://schemas.microsoft.com/search/local/ws/rest/v1',
     'callbackInSeconds': 30,
     'callbackUrl': 'https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrixA
syncCallback?key=Al3X4501N_LoywhAMfdbiDJe6Kd4cyPzon2xm55COHrQALhIX7Ay7qzBeSCa664
r&requestId=4d65bb0e-a5c1-4e94-b322-8fe755b2a08e&travelMode=Driving',
     'isAccepted': True,
     'isCompleted': False,
     'requestId': '4d65bb0e-a5c1-4e94-b322-8fe755b2a08e'}]}],
 'statusCode': 200,
 'statusDescription': 'OK',
 'traceId': 'b069ce80e345411bb283db920da4d967|MWH0032BEE|0.0.0.0|MWH0031C9B'}
```

[88]:
```
res = requests.get('https://dev.virtualearth.net/REST/v1/Routes/
  ↪DistanceMatrixAsyncCallback?
  ↪key=Al3X4501N_LoywhAMfdbiDJe6Kd4cyPzon2xm55COHrQALhIX7Ay7qzBeSCa664r&requestId=0248b9ba-d39
res.json()
```

[88]:
```
{'authenticationResultCode': 'ValidCredentials',
 'brandLogoUri': 'https://dev.virtualearth.net/Branding/logo_powered_by.png',
 'copyright': 'Copyright © 2023 Microsoft and its suppliers. All rights
reserved. This API cannot be accessed and the content and any results may not be
used, reproduced or transmitted in any manner without express written permission
from Microsoft Corporation.',
 'resourceSets': [{'estimatedTotal': 1,
   'resources': [{'__type':
'RouteProxyAsyncResult:http://schemas.microsoft.com/search/local/ws/rest/v1',
     'callbackInSeconds': -1,
     'callbackUrl': 'https://dev.virtualearth.net/REST/v1/Routes/DistanceMatrixA
syncCallback?key=Al3X4501N_LoywhAMfdbiDJe6Kd4cyPzon2xm55COHrQALhIX7Ay7qzBeSCa664
r&requestId=0248b9ba-d393-4c38-b704-b889d5417004&travelMode=Driving',
     'isAccepted': True,
     'isCompleted': True,
     'requestId': '0248b9ba-d393-4c38-b704-b889d5417004',
     'resultUrl': 'https://routematrixpremium.blob.core.windows.net/finalresults
/0248b9ba-d393-4c38-b704-b889d5417004'}]}],
 'statusCode': 200,
 'statusDescription': 'OK',
 'traceId': 'd617ee8bca9f4ecb98db3b451f538fff|MWH0032BEA|0.0.0.0'}
```

[89]:
```
result = requests.get("https://routematrixpremium.blob.core.windows.net/
  ↪finalresults/0248b9ba-d393-4c38-b704-b889d5417004")
json_data = result.content.decode('utf-8-sig')
data = json.loads(json_data)
data.keys()
```

[89]:
```
dict_keys(['isAccepted', 'isCompleted', 'requestId', 'callbackInSeconds',
'timeToService', 'origins', 'destinations', 'results'])
```

[90]:
```
df_travel = pd.json_normalize(data, "results")
df_travel.head()
```

[90]:

	originIndex	destinationIndex	travelDistance	travelDuration
0	0	0	6.634	20.1333
1	1	0	6.610	19.9500
2	2	0	3.649	13.2167
3	3	0	13.274	24.0167
4	4	0	1.792	6.2167

[91]:
```
df_travel.sort_values(by=['travelDistance'], inplace=True)
df_travel.head()
```

[91]:

	originIndex	destinationIndex	travelDistance	travelDuration
185	185	0	0.823	2.8167
88	88	0	1.004	4.2833
194	194	0	1.024	3.8167
275	275	0	1.025	3.8667
191	191	0	1.044	4.2167

[93]:
```
df_stations_sf.iloc[df_travel.index[0]]
```

[93]:
```
Station Name                              Lombard St at Columbus Ave
Latitude                                                   37.802767
Longitude                                                -122.413276
Address          708 Lombard St, San Francisco, CA 94133, Unite…
Name: 186, dtype: object
```

We should bike to the station called Lombard St at Columbus Ave. This is station differs from Part A because in Part A we used manhattan distances, distances between 2 points on a grid. However, using the Bing Maps API, we are using driving distances, which is the distance between 2 points on a map. This is why the results differ.

1.3 Submission Instructions

- After you have completed the notebook, select **Runtime > Run all**
- After the notebook finishes rerunning check to make sure that you have no errors and everything runs properly. Fix any problems and redo this step until it works.
- Rename this notebook by clicking on "DATA 301 Assignment 05 - YOUR NAMES HERE" at the very top of this page. Replace "YOUR NAMES HERE" with the first and last names of you and your partner (if you worked with one).
- Expand all cells with View > Expand Sections
- Save a PDF version: File > Print > Save as PDF
  - Under "More Settings" make sure "Background graphics" is checked
  - Printing Colab to PDF doesn't always work so well and some of your output might get cutoff. That's ok.
  - It's not necessary, but if you want a more nicely formatted PDF you can uncomment and run the code in the following cell. (Here's a video with other options.)
- Download the notebook: File > Download .ipynb
- Submit the notebook and PDF in Canvas. If you worked in a pair, only one person should submit in Canvas.

[ ]:
```
# !wget -nc https://raw.githubusercontent.com/brpy/colab-pdf/master/colab_pdf.py
# from colab_pdf import colab_pdf
# colab_pdf('DATA 301 Lab4A - YOUR NAMES HERE.ipynb')
```