DP0701EN-2-2-1-Foursquare-API-py-v1.0

June 21, 2019

Learning FourSquare API with Python

0.1 Introduction

In this lab, you will learn in details how to make calls to the Foursquare API for different purposes. You will learn how to construct a URL to send a request to the API to search for a specific type of venues, to explore a particular venue, to explore a Foursquare user, to explore a geographical location, and to get trending venues around a location. Also, you will learn how to use the visualization library, Folium, to visualize the results.

0.2 Table of Contents

0.2.1 Import necessary Libraries

In [ ]:
```
import requests # library to handle requests
import pandas as pd # library for data analsysis
import numpy as np # library to handle data in a vectorized manner
import random # library for random number generation

!conda install -c conda-forge geopy --yes
from geopy.geocoders import Nominatim # module to convert an address into latitude and longitude values

# libraries for displaying images
from IPython.display import Image
from IPython.core.display import HTML

# tranforming json file into a pandas dataframe library
from pandas.io.json import json_normalize
```

```
!conda install -c conda-forge folium=0.5.0 --yes
import folium # plotting library

print('Folium installed')
print('Libraries imported.')
```

0.2.2 Define Foursquare Credentials and Version

Make sure that you have created a Foursquare developer account and have your credentials handy

In [ ]:
```
CLIENT_ID = 'your-client-ID' # your Foursquare ID
CLIENT_SECRET = 'your-client-secret' # your Foursquare Secret
VERSION = '20180604'
LIMIT = 30
print('Your credentails:')
print('CLIENT_ID: ' + CLIENT_ID)
print('CLIENT_SECRET:' + CLIENT_SECRET)
```

Let's again assume that you are staying at the Conrad hotel. So let's start by converting the Contrad Hotel's address to its latitude and longitude coordinates. In order to define an instance of the geocoder, we need to define a user_agent. We will name our agent foursquare_agent, as shown below.

In [ ]:
```
address = '102 North End Ave, New York, NY'

geolocator = Nominatim(user_agent="foursquare_agent")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print(latitude, longitude)
```

0.3 1. Search for a specific venue category

https://api.foursquare.com/v2/venues/**search**?client_id=**CLIENT_ID**&client_secret=**CLIENT_SECRET**&

Now, let's assume that it is lunch time, and you are craving Italian food. So, let's define a query to search for Italian food that is within 500 metres from the Conrad Hotel.

In [ ]:
```
search_query = 'Italian'
radius = 500
print(search_query + ' .... OK!')
```

Define the corresponding URL

In [ ]:
```
url = 'https://api.foursquare.com/v2/venues/search?client_id={}&client_secret={}&ll={},{}&v={}&que
url
```

Send the GET Request and examine the results

In [ ]:
```
results = requests.get(url).json()
results
```

Get relevant part of JSON and transform it into a *pandas* dataframe

In [ ]:
```
# assign relevant part of JSON to venues
venues = results['response']['venues']

# tranform venues into a dataframe
dataframe = json_normalize(venues)
dataframe.head()
```

Define information of interest and filter dataframe

In [ ]:
```
# keep only columns that include venue name, and anything that is associated with location
filtered_columns = ['name', 'categories'] + [col for col in dataframe.columns if col.startswith('location.')] +
dataframe_filtered = dataframe.loc[:, filtered_columns]

# function that extracts the category of the venue
def get_category_type(row):
    try:
        categories_list = row['categories']
    except:
        categories_list = row['venue.categories']

    if len(categories_list) == 0:
        return None
    else:
        return categories_list[0]['name']

# filter the category for each row
dataframe_filtered['categories'] = dataframe_filtered.apply(get_category_type, axis=1)

# clean column names by keeping only last term
dataframe_filtered.columns = [column.split('.')[-1] for column in dataframe_filtered.columns]

dataframe_filtered
```

Let's visualize the Italian restaurants that are nearby

In [ ]:
```
dataframe_filtered.name
```

In [ ]:
```
venues_map = folium.Map(location=[latitude, longitude], zoom_start=13) # generate map centred aroun

# add a red circle marker to represent the Conrad Hotel
folium.features.CircleMarker(
    [latitude, longitude],
```

```
    radius=10,
    color='red',
    popup='Conrad Hotel',
    fill = True,
    fill_color = 'red',
    fill_opacity = 0.6
).add_to(venues_map)

# add the Italian restaurants as blue circle markers
for lat, lng, label in zip(dataframe_filtered.lat, dataframe_filtered.lng, dataframe_filtered.categories):
    folium.features.CircleMarker(
        [lat, lng],
        radius=5,
        color='blue',
        popup=label,
        fill = True,
        fill_color='blue',
        fill_opacity=0.6
    ).add_to(venues_map)

# display map
venues_map
```

0.4 2. Explore a Given Venue

https://api.foursquare.com/v2/venues/**VENUE_ID**?client_id=**CLIENT_ID**&client_secret=**CLIENT_SECR**

0.4.1 A. Let's explore the closest Italian restaurant -- *Harry's Italian Pizza Bar*

In [ ]:
```
venue_id = '4fa862b3e4b0ebff2f749f06' # ID of Harry's Italian Pizza Bar
url = 'https://api.foursquare.com/v2/venues/{}?client_id={}&client_secret={}&v={}'.format(venue_id,
url
```

Send GET request for result

In [ ]:
```
result = requests.get(url).json()
print(result['response']['venue'].keys())
result['response']['venue']
```

0.4.2 B. Get the venue's overall rating

In [ ]:
```
try:
    print(result['response']['venue']['rating'])
except:
    print('This venue has not been rated yet.')
```

That is not a very good rating. Let's check the rating of the second closest Italian restaurant.

In [ ]:
```
venue_id = '4f3232e219836c91c7bfde94' # ID of Conca Cucina Italian Restaurant
url = 'https://api.foursquare.com/v2/venues/{}?client_id={}&client_secret={}&v={}'.format(venue_id,

result = requests.get(url).json()
try:
    print(result['response']['venue']['rating'])
except:
    print('This venue has not been rated yet.')
```

Since this restaurant has no ratings, let's check the third restaurant.

In [ ]:
```
venue_id = '3fd66200f964a520f4e41ee3' # ID of Ecco
url = 'https://api.foursquare.com/v2/venues/{}?client_id={}&client_secret={}&v={}'.format(venue_id,

result = requests.get(url).json()
try:
    print(result['response']['venue']['rating'])
except:
    print('This venue has not been rated yet.')
```

Since this restaurant has a slightly better rating, let's explore it further.

0.4.3 C. Get the number of tips

In [ ]:
```
result['response']['venue']['tips']['count']
```

0.4.4 D. Get the venue's tips

https://api.foursquare.com/v2/venues/**VENUE_ID**/tips?client_id=**CLIENT_ID**&client_secret=**CLIENT_S**

Create URL and send GET request. Make sure to set limit to get all tips

In [ ]:
```
## Ecco Tips
limit = 15 # set limit to be greater than or equal to the total number of tips
url = 'https://api.foursquare.com/v2/venues/{}/tips?client_id={}&client_secret={}&v={}&limit={}'.for

results = requests.get(url).json()
results
```

Get tips and list of associated features

In [ ]:
```
tips = results['response']['tips']['items']

tip = results['response']['tips']['items'][0]
tip.keys()
```

Format column width and display all tips

In [ ]:
```
pd.set_option('display.max_colwidth', -1)

tips_df = json_normalize(tips) # json normalize tips

# columns to keep
filtered_columns = ['text', 'agreeCount', 'disagreeCount', 'id', 'user.firstName', 'user.lastName', 'user.gende
tips_filtered = tips_df.loc[:, filtered_columns]

# display tips
tips_filtered
```

Now remember that because we are using a personal developer account, then we can access only 2 of the restaurant's tips, instead of all 15 tips.

0.5 3. Search a Foursquare User

https://api.foursquare.com/v2/users/**USER_ID**?client_id=**CLIENT_ID**&client_secret=**CLIENT_SECRET**

0.5.1 Define URL, send GET request and display features associated with user

In [ ]:
```
user_id = '484542633' # user ID with most agree counts and complete profile

url = 'https://api.foursquare.com/v2/users/{}?client_id={}&client_secret={}&v={}'.format(user_id, CL

# send GET request
results = requests.get(url).json()
user_data = results['response']['user']

# display features associated with user
user_data.keys()
```

In [ ]:
```
print('First Name: ' + user_data['firstName'])
print('Last Name: ' + user_data['lastName'])
print('Home City: ' + user_data['homeCity'])
```

How many tips has this user submitted?

In [ ]:
```
user_data['tips']
```

Wow! So it turns out that Nick is a very active Foursquare user, with more than 250 tips.

0.5.2 Get User's tips

In [ ]:
```
# define tips URL
url = 'https://api.foursquare.com/v2/users/{}/tips?client_id={}&client_secret={}&v={}&limit={}'.form

# send GET request and get user's tips
results = requests.get(url).json()
```

```
tips = results['response']['tips']['items']

# format column width
pd.set_option('display.max_colwidth', -1)

tips_df = json_normalize(tips)

# filter columns
filtered_columns = ['text', 'agreeCount', 'disagreeCount', 'id']
tips_filtered = tips_df.loc[:, filtered_columns]

# display user's tips
tips_filtered
```

Let's get the venue for the tip with the greatest number of agree counts

```
In [ ]: tip_id = '5ab5575d73fe2516ad8f363b' # tip id

        # define URL
        url = 'http://api.foursquare.com/v2/tips/{}?client_id={}&client_secret={}&v={}'.format(tip_id, CLIEN

        # send GET Request and examine results
        result = requests.get(url).json()
        print(result['response']['tip']['venue']['name'])
        print(result['response']['tip']['venue']['location'])
```

0.5.3 Get User's friends

```
In [ ]: user_friends = json_normalize(user_data['friends']['groups'][0]['items'])
        user_friends
```

Interesting. Despite being very active, it turns out that Nick does not have any friends on Foursquare. This might definitely change in the future.

0.5.4 Retrieve the User's Profile Image

```
In [ ]: user_data
```

```
In [ ]: # 1. grab prefix of photo
        # 2. grab suffix of photo
        # 3. concatenate them using the image size
        Image(url='https://igx.4sqi.net/img/user/300x300/484542633_mK2Yum7T_7Tn9fWpndidJsmw2Hof_6T5
```

0.6 4. Explore a location

https://api.foursquare.com/v2/venues/**explore**?client_id=**CLIENT_ID**&client_secret=**CLIENT_SECRET**&

So, you just finished your gourmet dish at Ecco, and are just curious about the popular spots around the restaurant. In order to explore the area, let's start by getting the latitude and longitude values of Ecco Restaurant.

In [ ]:
```
latitude = 40.715337
longitude = -74.008848
```

Define URL

In [ ]:
```
url = 'https://api.foursquare.com/v2/venues/explore?client_id={}&client_secret={}&ll={},{}&v={}&rad
url
```

Send GET request and examine results

In [ ]:
```
import requests
```

In [ ]:
```
results = requests.get(url).json()
'There are {} around Ecco restaurant.'.format(len(results['response']['groups'][0]['items']))
```

Get relevant part of JSON

In [ ]:
```
items = results['response']['groups'][0]['items']
items[0]
```

Process JSON and convert it to a clean dataframe

In [ ]:
```
dataframe = json_normalize(items) # flatten JSON

    # filter columns
    filtered_columns = ['venue.name', 'venue.categories'] + [col for col in dataframe.columns if col.startswith('v
    dataframe_filtered = dataframe.loc[:, filtered_columns]

    # filter the category for each row
    dataframe_filtered['venue.categories'] = dataframe_filtered.apply(get_category_type, axis=1)

    # clean columns
    dataframe_filtered.columns = [col.split('.')[-1] for col in dataframe_filtered.columns]

    dataframe_filtered.head(10)
```

Let's visualize these items on the map around our location

In [ ]:
```
venues_map = folium.Map(location=[latitude, longitude], zoom_start=15) # generate map centred aroun


    # add Ecco as a red circle mark
    folium.features.CircleMarker(
        [latitude, longitude],
        radius=10,
```

```
        popup='Ecco',
        fill=True,
        color='red',
        fill_color='red',
        fill_opacity=0.6
        ).add_to(venues_map)


    # add popular spots to the map as blue circle markers
    for lat, lng, label in zip(dataframe_filtered.lat, dataframe_filtered.lng, dataframe_filtered.categories):
        folium.features.CircleMarker(
            [lat, lng],
            radius=5,
            popup=label,
            fill=True,
            color='blue',
            fill_color='blue',
            fill_opacity=0.6
            ).add_to(venues_map)

    # display map
    venues_map
```

0.7 5. Explore Trending Venues

https://api.foursquare.com/v2/venues/**trending**?client_id=**CLIENT_ID**&client_secret=**CLIENT_SECRET**

Now, instead of simply exploring the area around Ecco, you are interested in knowing the venues that are trending at the time you are done with your lunch, meaning the places with the highest foot traffic. So let's do that and get the trending venues around Ecco.

In [ ]:
```
# define URL
url = 'https://api.foursquare.com/v2/venues/trending?client_id={}&client_secret={}&ll={},{}&v={}'.fo

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

0.7.1 Check if any venues are trending at this time

In [ ]:
```
if len(results['response']['venues']) == 0:
    trending_venues_df = 'No trending venues are available at the moment!'

else:
    trending_venues = results['response']['venues']
    trending_venues_df = json_normalize(trending_venues)

    # filter columns
```

```
    columns_filtered = ['name', 'categories'] + ['location.distance', 'location.city', 'location.postalCode', 'loc
    trending_venues_df = trending_venues_df.loc[:, columns_filtered]

    # filter the category for each row
    trending_venues_df['categories'] = trending_venues_df.apply(get_category_type, axis=1)
```

In [ ]:
```
# display trending venues
trending_venues_df
```

Now, depending on when you run the above code, you might get different venues since the venues with the highest foot traffic are fetched live.

0.7.2 Visualize trending venues

In [ ]:
```
if len(results['response']['venues']) == 0:
    venues_map = 'Cannot generate visual as no trending venues are available at the moment!'

else:
    venues_map = folium.Map(location=[latitude, longitude], zoom_start=15) # generate map centred aro


    # add Ecco as a red circle mark
    folium.features.CircleMarker(
        [latitude, longitude],
        radius=10,
        popup='Ecco',
        fill=True,
        color='red',
        fill_color='red',
        fill_opacity=0.6
    ).add_to(venues_map)


    # add the trending venues as blue circle markers
    for lat, lng, label in zip(trending_venues_df['location.lat'], trending_venues_df['location.lng'], trending_
        folium.features.CircleMarker(
            [lat, lng],
            radius=5,
            poup=label,
            fill=True,
            color='blue',
            fill_color='blue',
            fill_opacity=0.6
        ).add_to(venues_map)
```

In [ ]:
```
# display map
venues_map
```

0.7.3 Thank you for completing this lab!

This notebook was created by Alex Aklson. I hope you found this lab interesting and educational. Feel free to contact me if you have any questions!

This notebook is part of a course on **Coursera** called *Applied Data Science Capstone*. If you accessed this notebook outside the course, you can take this course online by clicking here.