**Mining Facebook**

**Introduction**

When it comes to social media, aside from Twitter, Facebook is the reigning king. With over 650 million active users, the social media conglomerate has so much data coursing through its veins to the point that we could paint an infinite number of pictures with it. That being said, Python makes it remarkably painless and easy to access this data with only a few lines of code.

**Purpose**

The purpose of this exercise is to shine Python in the light of social media by using it to mine Facebook data.

**Before We Begin…**

Please, if you all run into any trouble during the course of this exercise, do not hesitate to seek help from me. I’m here for you!!! ☺

**Procedure**

**Part 1: Inspecting Facebook’s Graph API**

When it comes to programming, APIs (application-programming interfaces) are your best friend. In Layman’s terms, an API acts as a set of instructions that enable you to access a web tool or service. In our case, we will be using Facebook’s Graph API, which enables us to gather all sorts of data from you, your friends, other Facebook pages, etc. In order to access the Graph API, we first need to acquire an access token.

1. Navigate to <https://www.developers.facebook.com>.
2. On the developer’s homepage, click Register Now and follow the prompts. On the Tell us about you tab, make sure to check any programming languages you know (definitely Python). If it asks where you host and distribute your apps, you can select Apple App Store and Google Play even though we will not be creating an app for this exercise.
3. Once you have completed the registration process, navigate back to the developer homepage. On the menu bar, click on Tools and then click on Graph Explorer. This will take you to the Graph API Explorer page.
4. On the Graph API Explorer page, you should see a field titled Access Token. If there is nothing present in this field, click on Get Access Token. A window should pop up asking what permissions you would to have when making Graph API requests. I personally recommend checking all of the fields so that way we have full access to the Graph API. Do this for both the User Data Permissions and Extended Permissions. When finished, click Get Access Token.
5. The access token generated from doing step four will be key later on in this exercise when we start making requests to the Graph API from IPython. Below the Access Token field is the Graph API GUI (Graphic User Interface). Within this interface, you can test out different Graph API requests and see how the data requested looks like in a JSON format. As an example, lets get our Facebook id, name, and ten likes along with a list of ten friend’s ids, names, and ten likes per friend.
6. In the field that starts with Node: me, click on Search for a field and type friends. After doing so, a subfield will appear under friends. In the subfield, type limit. After pressing enter, set the limit number to ten. A new subfield should have popped up after typing limit and pressing enter. In the next subfield, type id. Then, in the next subfield, type name. Lastly, type likes in one last subfield. If you look in the field next to GET (right above the Node: me field), you should see something that looks like this…

fields=id,name,likes.limit(10),friends.limit(10).fields(id,name,likes.limit(10))

This represents all of the fields we are going to query. On the right hand side of the page, click the Submit button and you should see the JSON output update automatically. Note that we will not be able to see the list of friends we have requested until we make the request in IPython.

**Part 2: Querying the Facebook Graph API using IPython**

Now that we have become a Facebook developer and have explored the Graph API, let’s go ahead and establish a connection with Facebook in IPython so that we can make the requests we have mapped out in the Graph API Explorer online.

1. Open a Terminal/Command Prompt. Before we can query the Graph API, we first have to install a Python library called requests. This will enable us to make API requests over HTTP (Hyper Text Transfer Protocol) or, in layman’s terms, it allows us to use Python to access web content. To download the requests library, simply type pip install requests. Press enter.

1. After installing requests, run IPython. Now the first thing we need to do is import the requests library so we can make the API requests as well as the JSON library so we can format our data in the form of JSON. Type import requests followed by pressing enter and then type import json followed by pressing enter.
2. Now that we have imported the necessary libraries, we need to set up a connection with the Graph API. To do this, we will save the Graph API URL into a variable. Type graph\_url = ‘https://graph.facebook.com/me’. Press enter.
3. Now that we have the connection set up in the form of a variable, lets specify the information (fields) we want to retrieve from the Graph API. Remember that field next to GET that contained all of the fields we are going to query? Go ahead and paste that line into a variable titled fields. Remember to put single quotes around the contents of the variable.
4. In order to make our API request, we need to set our Access Token into a variable. Type ACCESS\_TOKEN = ‘*your\_access\_token\_goes\_here’*. Press enter.
5. Now that we have that set up, lets go ahead and put the whole request together. Type url = ‘%s?fields=%s&access\_token=%s’ % \ and the press enter. On the next line, type (graph\_url, fields, ACCESS\_TOKEN,). Press enter. In this case, the %s(s) on the first line are replaced with the respective fields specified on the second line.
6. At this point, the URL we need to make the API request is complete. To view the URL, you can type print url. Now, let’s go ahead and make the request to the Graph API and store the content into a variable. Type data = requests.get(url).json(). This will retrieve the requested contents of the URL and format it to JSON.
7. For the final portion of this section, let’s print out the contents we have received from the Graph API in the IPython environment. To do this, type print json.dumps(data, indent=1). It is important to note that if some of your friends have more stringent privacy settings, their bios may not be returned and thus not displayed.

**Part 3: Outputting the Contents Retrieved from the Graph API into a File**

One last thing we can do with the data we have collected from the Graph API is store it in a file for later use/manipulation. To do so, we will use the Python os library to set the directory for our outputted file.

1. In IPython type import os. On the next line, for Terminal, type os.chdir(’*your/directory/goes/here’)*. For Command Prompt, type os.chdir(‘your\\directory\\goes\\here’).
2. To check and see if your directory was changed successfully, you can type os.getcwd().
3. Now that we have our directory set, let’s go ahead and output the data we retrieved as a JSON file. For the purposes of this exercise, we will name the file fb\_data and use the.json extension.
4. Without further ado, type with open(‘fb\_data.json’,’w’) as outfile:. On the next line, type json.dump(data,outfile,indent=1). After completing this command, look in the directory you specified and you should see a file titled fb\_data.json. Open it up and make sure the data was successfully dumped into the file; if it wasn’t, the file will more than likely be blank.

**Wrap-Up**

Congratulations! You have just successfully mined Facebook using its Graph API. This exercise was only a taste as to what you can search for and retrieve using the Graph API. Have that being said; please feel free to mess around with it and find some awesome data to tell a story with.