Social Media Tracking Documentation

Cedric Tan - Roar Sports

April 2019

Abstract

The point of this documentation is to outline the details of the Python program that will track the social media following of Click Agency clients. The actual data is transported into an Excel file where independent analysis is conducted.

Contents

1 Introduction

This document aims to spell out the documentation of the Python programme that will automate social media tracking for Roar Sports. The program focuses on the key social media platforms for scraping follower data. Further analysis will be done with the excel content at a later stage. The following social media platforms are included in this documentation:

1. Facebook
2. Twitter
3. Twitch
4. Youtube
5. Instagram

Following this guide will help you understand how the program works.

2 Facebook

Beginning with the Facebook interface, the program will request access to the Facebook **Graph API** by authenticating with the website, making a request to access for the data of a particular page or profile and subsequently saving this profile data to a database where we can print it all out on excel.

3 Twitter

With Twitter, the method will utilise **Tweepy** which is a library that allows for quick access to the Twitter API.

3.1 Code Rundown

First we will begin with the dependencies required to gain the data:

```
1
2 from tweepy import OAuthHandler
3 from tweepy import API
4 from tweepy import Cursor
5 import sys
```

Here in the code above, you can see the rundown on each dependency:

- Tweepy is a Python library that we can use to access the Twitter API
- OAuthHandler will allow us to painlessly authorise access to get data from the API itself
- API is the main access point for Tweepy to get the data
- Cursor allows for pagination and access to large volumes of data that we might need to segment into sections i.e. pages
- Sys provides some basic functionality for the interpreter

Then we will set up the keys to authorise our access to the Twitter API:

```
1
2 # These are the keys from the app created on the
    developer account
3
4 consumer_key = "..."
5 consumer_secret = "..."
6 access_token = "..."
7 access_secret_token = "..."
8
9 # This begins the authorisation process with the
    Twitter API so that the program can authenticate it
     self with the website
10
11 auth = OAuthHandler(consumer_key, consumer_secret)
```

```
auth.set_access_token(access_token, access_secret_token)
auth_api = API(auth)
```

Then with all parts authorised, we can begin to get the data from the API itself through simple get commands.

```

# We will first get the Twitter handle of the account that we want to check out:

user = auth_api.get.user("barackobama")

# Having gotten the object with the get.user function, in this case Barack Obama's account, we can execute some functions on it like:

screen_name = user.screen_name # Take the user's screen name
follower_count = user.followers_count # Take the user's follower count

# Then printing these, we return values specific to the account

print (screen_name)
print (follower_count)

# This returns:
  BarackObama
  105699616 # 17/04/2019 12:07pm
```

3.2 Data Retrieval

From the previous section, we can just iterate the code over and over again from our database of Twitter Handles. This can either be hard coded into the system or taken from an Excel and imported in.

The iteration of code will be a simple for loop such that the output will show the each line of data specific to the Twitter handle. Taking the code from above, we can implement the loop below:

```
# Defining a dictionary
name_dictionary = ["name1", "name2", "name3", "name4"]

for name in name_dictionary:
```

```
  user = auth_api.get.user(name)
  screen_name = user.screen_name
  follower_count = user.followers_count

  print(screen.name)
  print(follower.count)

# This will return
  name1
  name1_follower_count
  ...
  name4
  name4_follower_count
```

4 Twitch

Twitch has API authentication through your Twitch account with the same OAuth processes and application interfaces that are necessary for the API request.

4.1 Code Rundown

5 Youtube

The Google developer console makes the Youtube Data API very accessible - all you require is a Google account to get an API key that is valid once you activate the Youtube Data API package on your account.

5.1 Code Rundown

Again we will begin with the dependencies that we are looking to use. For Youtube we are simply going to request some data from the Youtube Data API **without using a particular software development kit** which makes it more simplistic, requiring less installation.

```
1
2 import urllib.request
3 import json
4
5 key = "..."
```

The rundown:

- Urllib.request is the library used to draw requests from the actual API
- JSON allows us to read these objects and draw the important information we want from them
- The key is your personal authentication to gain access to the API

Then we can call on channel handles to get information on the amount of subscribers they have using the JSON object we have called:

```
1
2 dictionary = ["pewdiepie","bgfilms", "rhettandlink2"]
3
4 for name in dictionary:
5   # Requesting access to the API along with utilising
      the key to gain access
6
7     data = urllib.request.urlopen("https://www.
      googleapis.com/youtube/v3/channels?part=statistics&
      forUsername="+name+"&key="+key).read()
8
9     # Loading the JSON requested data and then
      searching for the subscriber count
```

```
    subs = json.loads(data)["items"][0]["statistics"]["
    subscriberCount"]

    # Printing the output
    print(name + " has " + "{:,d}".format(int(subs)) +
    " subscribers")

    # This will give us:

    pewdiepie has 94,317,649 subscribers
  bgfilms has 4,122,055 subscribers
  rhettandlink2 has 15,294,165 subscribers
```

6 Object Oriented Structure

Having discussed the various APIs that we are going to use, the next step is to discuss what type of structure we will use to access the required information across the board. Probably the most scalable and efficient structure to use with the ability to call numerous functions from would be using classes.

The structure of a class provides a mean of bundling data and functionality together. Creating a class is creating a new type of object allowing *instances of that object* to be created again and again.

For the purposes of Social Media Tracking and Performance, we will create the influencer class which stores the data of each influencer in a new instance. From there, we can call functions as necessary on these influencers to gain information on their numbers through our APIs.

6.1 Code Rundown

Setting up the class is easy and we will give it a lot of variables related to the data that we want to collect.

```

class influencer:
  def __init__(self, name, gender, gamer_tag, fb_tag,
    twitter_tag, yt_tag, insta_tag, twitch_tag)
  self.name = name
  self.gender = gender
  self.gamer_tag = gamer_tag
  self.fb_tag = fb_tag # Facebook Take
```

```
8   self.twitter_tag = twitter_tag # Twitter Tag
9   self.yt_tag = yt_tag # Youtube Channel Tag
10  self.insta_tag = insta_tag # Instagram Tag
11  self.twitch_tag = twitch_tag # Twitch Channel Tag
```

From this, we can recognise the different social media accounts assigned to each along with some other basic information such as name and gender.

Following that, we can input some basic variables to store for the future

7 Data Structures