Downloading Tweets – Take II

The goal of this post is to walk you through a Python script designed to download tweets by a set of Twitter users and insert them into an SQLite database.

In a previous post I supplied a brief, temporary attempt at providing an overview of how to download tweets sent by a list of Twitter users – but I ended the post by pointing pointing people to a good tutorial written by my former PhD student and now co-author Wayne Xu.

Now I am finally getting around to posting my own tutorial on how to download tweets sent by a list of different Twitter users. It is directed at those who are new to Python and/or downloading data from the Twitter API. We will be using Python to download the tweets and will be inserting the tweets into an SQLite database. This code will allow you to download up to the latest 3,200 tweets sent by each Twitter user. I will not go over the script line-by-line but will instead attempt to provide you a 'high-level' understanding of what we are doing – just enough so that you can run the script successfully yourself.

Before running this script, you will need to:

- Have Anaconda Python 2.7 installed
- Have your Twitter API details handy
- Have created a CSV file (e.g., in Excel) containing the Twitter handles you wish to download. Below is a sample you can download and use for this tutorial. Name it *accounts.csv* and place it in the same directory as the Python script.

https://gist.github.com/gdsaxton/1825a5d455e61732eff69dc8cc17dd59

If you are completely new to Python and the Twitter API, you

should first make your way through the following tutorials, which will help you get set up and working with Python:

- Which version of Python to download
- Running your first code
- Four ways to run your code
- Setting up access to the Twitter API

Another detailed tutorial I have created, Python Code Tutorial, is intended to serve as an introduction to how to access the Twitter API and then read the JSON data that is returned. It will be helpful for understanding what we're doing in this script.

At the end of this post I'll show the entire script. For now, I'll go over it in sections. The code is divided into six parts:

Part I: Overview and Importing Necessary Python Packages

The first line in the code is the shebang – you'll find this in all Python code.

Lines 4 – 24 contain the docstring – also a Python convention. This is a multi-line comment that describes the code. For single-line comments, use the # symbol at the start of the line.

In lines 29 – 44 we'll import some Python packages needed to run the code. Twython and simplejson can be installed by opening your Terminal and installing by entering pip install simplejson and then pip install Twython. For more details on

this process see this blog post.

Part II: Import Twython and Twitter App Key and Access Token

Lines 50-55 is where you will enter your Twitter *App Key* and *Access Token* (lines 53-54). If you have yet to do this you can refer to the tutorial on Setting up access to the Twitter API.

Part III: Define a Function for Getting Twitter Data

In this block of code we are creating a Python *function*. The function sets up which part of the Twitter API we wish to access (specifically, it is the get user timeline API), the number of tweets we want to get per page (I have chosen the maximum of 200), and whether we want to include retweets. We will call this function later on in the code.

Part IV: Set up SQLite Database Tables

Lines 85-255 are where you set up columns for the database to allow use of SQLite and SQLAlchemy. For the reasons why we're using SQLite you can refer to this post.

This is a very long, mechanical block of code. We are defining two *tables* for our SQLite database – one for the tweets and one for the Twitter account names we want to download. Within each table we are defining variable names and variable types. That is, we are naming every variable (column) we wish to create and saying whether it is a *string* variable (i.e., text) or an *integer* variable.

Why do we need to do this? We don't actually have to go into such detail, but doing this 'set-up' work now will make managing and analyzing the data later easier. When I use

SQLite I specifically like to use SQLAlchemy (we imported it earlier), which makes working with an SQLite database even easier. Though it is long, it is a standard piece of code that you can cut-and-paste into any tweet download script. I usually split this block of code into a separate file and then import it, but I'm including all code here in a single script to make it easier to understand all of the moving parts.

Part V: Write function for parsing and storing data returned by Twitter API

In Lines 263-554 we are writing a long function to help *write* the data to the SQLite database. At this point in the code we have still not downloaded any data. We are just setting up a function that we'll call later.

I won't go into all of the details of what we're doing here. The 'big picture' is that with this function we are looping over all 200 tweets in each page. For each tweet, we are grabbing certain bits of information returned by the Twitter API and then assigning those bits of information to the database variables we set up earlier.

Now, a key step to this is understanding the data that are returned by the API. As is increasingly common with Web data, this API call returns data in JSON format. Behind the scenes, Python has grabbed a JSON file, which will have data on 200 tweets (one page of tweets). Each tweet is an object in the JSON file, and we are looping over each object. If you're interested in more details, the code we're using here plugs into the user_timeline part of the Twitter API; follow the link for a description of what the API does. You can also go here to see a list of the definitions for the variables returned by the API. To understand how this relates to JSON, you can take a look at my earlier post on downloading Twitter user data.

I'll give you two examples here: In line 279 we are taking the data included in the object's ['id'] variable and assigning that to a variable called *tweet_id*. In line 358 we are creating a variable called *num_characters* whose value is the number of characters in the tweet. We are doing this for every variable we have set up in Part IV earlier.

After we've created these variables, we now need to write them to our database. In lines 534-547 we tell our SQLite database that we want to update our TWEET database and which variables to include. You'll see these are the same variable names we used in Part IV. Line 549 contains the command to add the tweet data to our database.

Here comes a key part: lines 550-554 contain a *try…except* loop designed to catch duplicate tweets. Simply put, if the tweet already exists in the database it will skip over it. This is really important and one of the best reasons to use a database for downloading tweets. If you wanted, you could simply download your tweets to an Excel spreadsheet. However, each time you ran the script you would likely be downloading a truckload of duplicates. You don't want that.

The key to this lies in how we set up our TWEET database earlier; specifically, line 99 contains a *unique* constraint.

There can only be one entry with a given tweet_id_str value. Given that every tweet has a unique tweet ID, this is the best variable on which to make a unique constraint.

Part VI: Main Loop: Loop over each of the Twitter handles in the *Accounts* table and Download Tweets.

In lines 560-627 we are at the last step. In this block the first important thing we do is name our SQLite database on line 579.

Then in line 571 we query the *ACCOUNT* table in our database and assign that to a variable *all_ids*. This will work for the second time we run the script and all subsequent times.

The first time we run the script, though, our *ACCOUNT* table will be empty. Lines 576-583 check whether it is empty and, if so, will grab the details from our *accounts.csv* file and insert it into our database.

At line 586 we then begin a *for loop* – we will be looping over each Twitter ID (as indicated by the *Twitter_handle* variable in our *ACCOUNT* database).

Note that within this *for loop* we have a *while loop* (lines 596-618). What we are doing here is, for each Twitter ID, we are grabbing up to 16 pages' worth of tweets; this is the maximum allowed for by the Twitter API. It is in this loop that we actually call our functions. On line 598 we invoke our *get_data_user_timeline_all_pages* function, which on the first loop will grab page 1 for the Twitter ID, then page 2, then page 3, …. up to page 16 so long as there are data to return. Line 606 invokes the *write_data* function for each page.

Now let's put the whole thing together. To recap, what this entire script does is to loop over each of the Twitter accounts in the *Accounts* table of our database – and for each one it will grab up to 3,200 tweets and insert the tweets into an SQLite database.

Below is the entire script – download it and save it as *tweets.py* (or something similar) in the same directory as your *accounts.csv* file. Add in your Twitter API account details and you'll be good to go! For a refresher on the different ways you can run the script see this earlier post.

If you've found this post helpful please share on your favorite social media site.

You’re on your way to downloading your own Twitter data! Happy coding!

https://gist.github.com/gdsaxton/b0d36c10bbdb80e26b692a1d1a3e11de