Twitter API tutorial

by Wei Xu (http://www.cis.upenn.edu/~xwe/) Follow @cocoweixu **(Ohio State University)**

last updated Feb 28, 2016; originally written July 1, 2015

1. Getting Twitter API keys

To start with, you will need to have a Twitter account and obtain credentials (i.e. API key, API secret, Access token and Access token secret) on the Twitter developer site to access the Twitter API, following these steps:

- Create a Twitter user account if you do not already have one.
- Go to https://apps.twitter.com/ (https://apps.twitter.com/) and log in with your Twitter user account. This step gives you a Twitter dev account under the same name as your user account.
- Click “Create New App”
- Fill out the form, agree to the terms, and click “Create your Twitter application”
- In the next page, click on “Keys and Access Tokens” tab, and copy your “API key” and “API secret”. Scroll down and click “Create my access token”, and copy your “Access token” and “Access token secret”.

2. Installing a Twitter library

We will be using a Python library called Python Twitter Tools (https://pypi.python.org/pypi/twitter) to connect to Twitter API and downloading the data from Twitter. There are many other libraries (https://dev.twitter.com/overview/api/twitter-libraries>) in various programming languages that let you use Twitter API. We choose the Python Twitter Tools for this tutorial, because it is simple to use yet fully supports the Twitter API.

Download the Python Twitter tools at https://pypi.python.org/pypi/twitter (https://pypi.python.org/pypi/twitter).

Install the Python Twitter Tools package by typing in commands:

```
$ python setup.py --help
$ python setup.py build
$ python setup.py install
```

3. Connecting to Twitter Streaming APIs

The Streaming APIs give access to (usually a sample of) all tweets as they published on Twitter. On average, about 6,000 tweets per second are posted on Twitter and you (normal dev users) will get a small proportion (<=1%) of it. The Streaming APIs are one of the two types of Twitter APIs. The other one called REST APIs (we will talk about later in this tutorial), which is more suitable for singular searches, such as searching historic tweets, reading user profile information, or posting Tweets. The Streaming API **only** sends out real-time tweets, while the Search API (one of the popular REST APIs) gives historical tweets up to about a week with a max of a couple of hundreds. You may request elevated access (e.g. Firehose, Retweet, Link, Birddog or Shadow) for more data by contacting Twitter’s API support.

Basic Uses of Streaming APIs

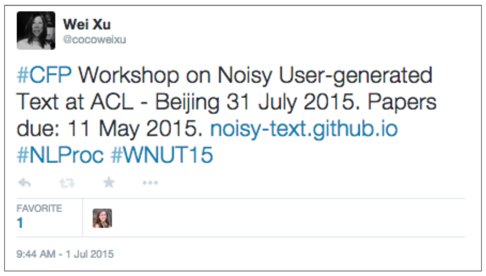
Create a file called *twitter_streaming.py*, and copy the code below into it. Make sure to enter your credentials obtained in the Step 1 above into ACCESS_TOKEN, ACCESS_SECRET, CONSUMER_KEY, and CONSUMER_SECRET.

```
# Import the necessary package to process data in JSON format
try:
    import json
except ImportError:
    import simplejson as json

# Import the necessary methods from "twitter" library
from twitter import Twitter, OAuth, TwitterHTTPError, TwitterStream

# Variables that contains the user credentials to access Twitter API
ACCESS_TOKEN = 'YOUR ACCESS TOKEN"'
ACCESS_SECRET = 'YOUR ACCESS TOKEN SECRET'
CONSUMER_KEY = 'YOUR API KEY'
CONSUMER_SECRET = 'ENTER YOUR API SECRET'

oauth = OAuth(ACCESS_TOKEN, ACCESS_SECRET, CONSUMER_KEY, CONSUMER_SECRET)

# Initiate the connection to Twitter Streaming API
twitter_stream = TwitterStream(auth=oauth)

# Get a sample of the public data following through Twitter
iterator = twitter_stream.statuses.sample()

# Print each tweet in the stream to the screen
# Here we set it to stop after getting 1000 tweets.
# You don't have to set it to stop, but can continue running
# the Twitter API to collect data for days or even longer.
tweet_count = 1000
for tweet in iterator:
    tweet_count -= 1
    # Twitter Python Tool wraps the data returned by Twitter
    # as a TwitterDictResponse object.
    # We convert it back to the JSON format to print/score
    print json.dumps(tweet)

    # The command below will do pretty printing for JSON data, try it out
    # print json.dumps(tweet, indent=4)

    if tweet_count <= 0:
        break
```

If you run the program by typing in the command:

```
$ python twitter_streaming.py
```

You will see tweets keep flowing in your screen. They are a sample of public data (including tweets and also events) flowing though Twitter at the moment. The data returned is in JSON (https://en.wikipedia.org/wiki/JSON) format. It may looks too much for now; it will become clearer in the next step how to read and process this data. Below is one example tweet:


Wei Xu
@cocoweixu

#CFP Workshop on Noisy User-generated Text at ACL - Beijing 31 July 2015. Papers due: 11 May 2015. noisy-text.github.io #NLProc #WNUT15

and its JSON format (often output without line breaks to save space but difficult to read and make sense of — but you can do pretty printing to include line breaks to make it more readable as shown in Step 5) :

```
{"favorited": false, "contributors": null, "truncated": false, "text": "#CFP Workshop on Noisy User-generated Text at ACL - Beijing 31 July 2015. Papers du
e: 11 May 2015. http://t.co/rcygyEowqH   #NLProc #WNUT15", "possibly_sensitive": false, "in_reply_to_status_id": null, "user": {"follow_request_sent": null,
 "profile_use_background_image": true, "default_profile_image": false, "id": 237918251, "verified": false, "profile_image_url_https": "https://pbs.twimg.co
m/profile_images/527088456967544832/DnclpoZO_normal.jpeg", "profile_sidebar_fill_color": "DDEEF6", "profile_text_color": "333333", "followers_count": 226,
"profile_sidebar_border_color": "C0DEED", "id_str": "237918251", "profile_background_color": "C0DEED", "listed_count": 13, "profile_background_image_url_htt
ps": "https://abs.twimg.com/images/themes/theme1/bg.png", "utc_offset": null, "statuses_count": 120, "description": "I am a postdoctoral researcher @PennCI
S, studying Natural Language Processing and Social Media.", "friends_count": 166, "location": "Philadelphia PA", "profile_link_color": "0084B4", "profile_im
age_url": "http://pbs.twimg.com/profile_images/527088456967544832/DnclpoZO_normal.jpeg", "following": null, "geo_enabled": true, "profile_background_image_u
rl": "http://abs.twimg.com/images/themes/theme1/bg.png", "name": "Wei Xu", "lang": "en", "profile_background_tile": false, "favourites_count": 88, "screen_n
ame": "cocoweixu", "notifications": null, "url": "http://www.cis.upenn.edu/~xwe/", "created_at": "Thu Jan 13 23:15:12 +0000 2011", "contributors_enabled": f
alse, "time_zone": null, "protected": false, "default_profile": true, "is_translator": false}, "filter_level": "low", "geo": null, "id": 616333141884674048,
 "favorite_count": 0, "lang": "en", "entities": {"user_mentions": [], "symbols": [], "trends": [], "hashtags": [{"indices": [0, 4], "text": "CFP"}, {"indice
s": [124, 131], "text": "NLProc"}, {"indices": [132, 139], "text": "WNUT15"}], "urls": [{"url": "http://t.co/rcygyEowqH", "indices": [99, 121], "expanded_ur
l": "http://noisy-text.github.io", "display_url": "noisy-text.github.io"}]}, "in_reply_to_user_id_str": null, "retweeted": false, "coordinates": null, "time
stamp_ms": "1435780246598", "source": "<a href=\"http://twitter.com\" rel=\"nofollow\">Twitter Web Client</a>", "in_reply_to_status_id_str": null, "in_reply
_to_screen_name": null, "id_str": "616333141884674048", "place": null, "retweet_count": 0, "created_at": "Wed Jul 01 19:50:46 +0000 2015", "in_reply_to_user
_id": null}
```

You can run the program and save the data into a file for analysis later using the following commend:

```
$ python twitter_streaming.py > twitter_stream_1000tweets.txt
```

Advanced Uses of Streaming APIs

The streaming API provides more advanced functions.

First, you can set different parameters (see here (https://dev.twitter.com/streaming/overview/request-parameters) for a complete list) to define what data to request. For example, you can track certain tweets by specifying keywords or location or language etc. The following code will get the tweets in English that include the term "Google":

```
iterator = twitter_stream.statuses.filter(track="Google", language="en")
```

Location is a bit tricky. Read here (https://dev.twitter.com/streaming/overview/request-parameters#locations) for a simple guide, and here (http://thoughtfaucet.com/search-twitter-by-location/) for a complete guide. Find tweets by location can be done either by the Streaming API (only geolocated tweets) or the Search API (user's location field is also used).

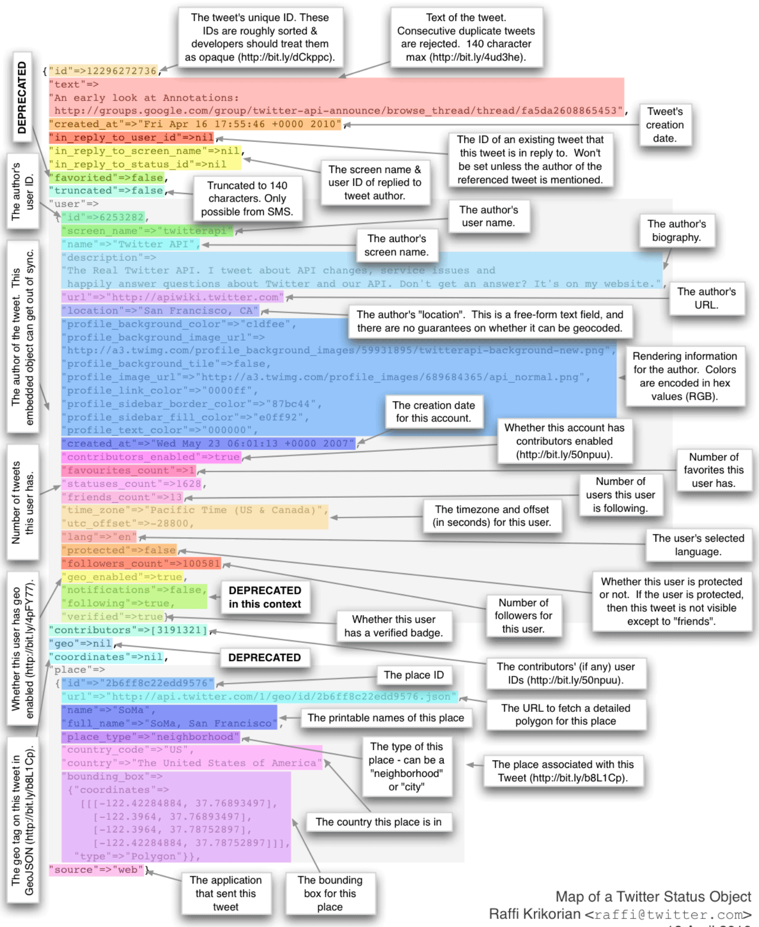
Second, by default, streaming API is connecting to the "public streams" — all public data on Twitter as we showed in the above example. Also, there are "user streams" and "site streams" that contains the data specific to the authenticated user or users (see here (https://dev.twitter.com/streaming/overview) for more details). For conducting research on Twitter data, you usually only need to use "public streams" to collect data. In case you do need to use other streams, here is how to specify it:

```
twitter_userstream = TwitterStream(auth=oauth, domain='userstream.twitter.com')
```

4. Reading and Processing Tweets in JSON format

The streaming API returns tweets (https://dev.twitter.com/overview/api/tweets), as well as several other types of messages (https://dev.twitter.com/streaming/overview/messages-types) (e.g. a tweet deletion notice, user update profile notice, etc), all in JSON format. Here we demonstrate how to read and process tweets in details. Other data in JSON format can be processed similarly.

Tweets, also known more generically as "status updates". This map made by Raffi Krikorian explains a tweet in JSON format:



```
{"id"=>12296272736,
 "text"=>
 "An early look at Annotations:
  http://groups.google.com/group/twitter-api-announce/browse_thread/thread/fa5da2608865453",
 "created_at"=>"Fri Apr 16 17:55:46 +0000 2010",
 "in_reply_to_user_id"=>nil,
 "in_reply_to_screen_name"=>nil,
 "in_reply_to_status_id"=>nil
 "favorited"=>false,
 "truncated"=>false,
 "user"=>
  {"id"=>6253282,
   "screen_name"=>"twitterapi",
   "name"=>"Twitter API",
   "description"=>
   "The Real Twitter API. I tweet about API changes, service issues and
    happily answer questions about Twitter and our API. Don't get an answer? It's on my website.",
   "url"=>"http://apiwiki.twitter.com",
   "location"=>"San Francisco, CA",
   "profile_background_color"=>"c1dfee",
   "profile_background_image_url"=>
   "http://a3.twimg.com/profile_background_images/59931895/twitterapi-background-new.png",
   "profile_background_tile"=>false,
   "profile_image_url"=>"http://a3.twimg.com/profile_images/689684365/api_normal.png",
   "profile_link_color"=>"0000ff",
   "profile_sidebar_border_color"=>"87bc44",
   "profile_sidebar_fill_color"=>"e0ff92",
   "profile_text_color"=>"000000",
   "created_at"=>"Wed May 23 06:01:13 +0000 2007",
   "contributors_enabled"=>true
   "favourites_count"=>1
   "statuses_count"=>1628,
   "friends_count"=>13
   "time_zone"=>"Pacific Time (US & Canada)",
   "utc_offset"=>-28800,
   "lang"=>"en"
   "protected"=>false
   "followers_count"=>100581
   "geo_enabled"=>true,
   "notifications"=>false,
   "following"=>true,
   "verified"=>true},
 "contributors"=>[3191321],
 "geo"=>nil,
 "coordinates"=>nil,
 "place"=>
  {"id"=>"2b6ff8c22edd9576",
   "url"=>"http://api.twitter.com/1/geo/id/2b6ff8c22edd9576.json"
   "name"=>"SoMa",
   "full_name"=>"SoMa, San Francisco",
   "place_type"=>"neighborhood"
   "country_code"=>"US",
   "country"=>"The United States of America"
   "bounding_box"=>
    {"coordinates"=>
      [[[-122.42284884, 37.76893497],
        [-122.3964, 37.76893497],
        [-122.3964, 37.78752897],
        [-122.42284884, 37.78752897]]],
     "type"=>"Polygon"}},
 "source"=>"web"}
```

The tweet's unique ID. These IDs are roughly sorted & developers should treat them as opaque (http://bit.ly/dCkppc).
Text of the tweet. Consecutive duplicate tweets are rejected. 140 character max (http://bit.ly/4ud3he).
DEPRECATED
Tweet's creation date.
The ID of an existing tweet that this tweet is in reply to. Won't be set unless the author of the referenced tweet is mentioned.
The screen name & user ID of replied to tweet author.
Truncated to 140 characters. Only possible from SMS.
The author's user ID.
The author's user name.
The author's screen name.
The author's biography.
The author of the tweet. This embedded object can get out of sync.
The author's URL.
The author's "location". This is a free-form text field, and there are no guarantees on whether it can be geocoded.
Rendering information for the author. Colors are encoded in hex values (RGB).
The creation date for this account.
Whether this account has contributors enabled (http://bit.ly/50npuu).
Number of favorites this user has.
Number of tweets this user has.
Number of users this user is following.
The timezone and offset (in seconds) for this user.
The user's selected language.
Whether this user is protected or not. If the user is protected, then this tweet is not visible except to "friends".
Number of followers for this user.
DEPRECATED in this context
Whether this user has a verified badge.
Whether this user has geo enabled (http://bit.ly/4pFY77).
The contributors' (if any) user IDs (http://bit.ly/50npuu).
DEPRECATED
The place ID
The URL to fetch a detailed polygon for this place
The printable names of this place
The type of this place - can be a "neighborhood" or "city"
The place associated with this Tweet (http://bit.ly/b8L1Cp).
The country this place is in
The geo tag on this tweet in GeoJSON (http://bit.ly/b8L1Cp).
The application that sent this tweet
The bounding box for this place

Map of a Twitter Status Object
Raffi Krikorian <raffi@twitter.com>
18 April 2010

Although this map is from 2010 and a bit out-of-date, it is a good visualization of tweet's JSON format. You can find the up-to-date information of tweet's format here (https://dev.twitter.com/overview/api/tweets).

Use Python library *json* or *simplejson* to read in the data in JSON format and process them:

```
# Import the necessary package to process data in JSON format
try:
    import json
except ImportError:
    import simplejson as json

# We use the file saved from last step as example
tweets_filename = 'twitter_stream_1000tweets.txt'
tweets_file = open(tweets_filename, "r")

for line in tweets_file:
    try:
        # Read in one line of the file, convert it into a json object
        tweet = json.loads(line.strip())
        if 'text' in tweet: # only messages contains 'text' field is a tweet
            print tweet['id'] # This is the tweet's id
            print tweet['created_at'] # when the tweet posted
            print tweet['text'] # content of the tweet

            print tweet['user']['id'] # id of the user who posted the tweet
            print tweet['user']['name'] # name of the user, e.g. "Wei Xu"
            print tweet['user']['screen_name'] # name of the user account, e.g. "cocoweixu"

            hashtags = []
            for hashtag in tweet['entities']['hashtags']:
                hashtags.append(hashtag['text'])
            print hashtags

    except:
        # read in a line is not in JSON format (sometimes error occured)
        continue
```

If, instead of the file *twitter_stream_1000tweets.txt*, use the example tweet in JSON format in the Streaming API section, this piece of code will output the following results:

```
616333141884674048
Wed Jul 01 19:50:46 +0000 2015
#CFP Workshop on Noisy User-generated Text at ACL - Beijing 31 July 2015. Papers due: 11 May 2015. http://t.co/rcygyEowqH   #NLProc #WNUT15
237918251
Wei Xu
cocoweixu
[u'CFP', u'NLProc', u'WNUT15']
```

Note that the same url will have a few different versions in the Twitter stream: *http://t.co/rcygyEowqH* in the text, *http://noisy-text.github.io* as the expanded full version, *noisy-text.github.io* as the display version.

For long-term data collection, you can setup a cron job (https://en.wikipedia.org/wiki/Cron). If you are interested in running a long term collection of one or multiple streaming queries, consider using Mark Dredze (http://www.cs.jhu.edu/~mdredze/)'s Twitter streaming library (https://github.com/mdredze/twitter_stream_downloader). This library wraps the basic streaming API with several helpful features, such as organizing data into files by data, support for multiple feed types, and ensuring feeds remain active after interruptions. You can run this library inside of crontab or supervisord.

5. Using Twitter Search API and Trends API

Besides the streaming APIs, Twitter also provide another type of APIs — REST APIs. It provides two main functionalities: *GET* data from Twitter and *POST* data (e.g. a tweet from your account) to Twitter. In this tutorial, we will demonstrate three most useful APIs to collect data for social media research: Search (tweets contain certain words), Trends (trending topics) and User (a user's tweets, followers, friends, etc.). For explanations of these key types of data offered by Twitter, see the lecture slides (syllabus.html) on this course website.

Search API

Similar to the Streaming API, you first import necessary Python packages and OAuth credentials as in Step 2. Then you can use search API like follows:

```
# Initiate the connection to Twitter REST API
twitter = Twitter(auth=oauth)

# Search for latest tweets about "#nlproc"
twitter.search.tweets(q='#nlproc')
```

Alternatively, you can search with more parameters (see a full list here (https://dev.twitter.com/rest/public/search)). For example, search for 10 latest tweets about "#nlproc" in English:

```
twitter.search.tweets(q='#nlproc', result_type='recent', lang='en', count=10)
```

Trends API

Twitter provides global trends and as well as localized tweets. The easiest and best way to see what trends are available and the place ids (which you will need to know to query localized trends or tweets), is by using this commend to request worldwide trends:

```
# Get all the locations where Twitter provides trends service
world_trends = twitter.trends.available(_woeid=1)
```

It returns all the trends that are offered by Twitter at the time. Here is part of the returned results:

```
{"name": "United States", "countryCode": "US", "url": "http://where.yahooapis.com/v1/place/23424977", "country": "United States", "parentid": 1, "placeTyp
e": {"code": 12, "name": "Country"}, "woeid": 23424977}

{"name": "San Francisco", "countryCode": "US", "url": "http://where.yahooapis.com/v1/place/2487956", "country": "United States", "parentid": 23424977, "plac
eType": {"code": 7, "name": "Town"}, "woeid": 2487956}

{"name": "Bangkok", "countryCode": "TH", "url": "http://where.yahooapis.com/v1/place/1225448", "country": "Thailand", "parentid": 23424960, "placeType": {"c
ode": 7, "name": "Town"}, "woeid": 1225448}
```

The places ids are WOEIDs (Where On Earth ID), which are 32-bit identifiers provided by Yahoo! GeoPlanet (https://developer.yahoo.com/geo/geoplanet/guide/concepts.html) project. And yes! Twitter is very international.

After you know the ids for the places you are interested in, you can get the local trends like this:

```
# Get all (it's always 10) trending topics in San Francisco (its WOEID is 2487956)
sfo_trends = twitter.trends.place(_id = 2487956)
```

The trends will be returned in JSON, again. This time we reformat JSON data in a prettier way to make it easier to read by human beings:

```
print json.dumps(sfo_trends, indent=4))
```

and get:

```
{
   "created_at":"2015-07-01T22:09:55Z",
   "trends":[
      {
         "url":"http://twitter.com/search?q=%23LiesIveToldMyParents",
         "query":"%23LiesIveToldMyParents",
         "name":"#LiesIveToldMyParents",
         "promoted_content":null
      },
      {
         "url":"http://twitter.com/search?q=%22Kevin+Love%22",
         "query":"%22Kevin+Love%22",
         "name":"Kevin Love",
         "promoted_content":null
      },

      ... [and another 8 trends omitted here to save space]

}
```

If you want to get the real tweets in each trend, use the Search API to get them.

How often do the Twitter trending topics change? It is not disclosed by Twitter, but based on my experience, you will get most of them by querying every 5 minutes.

User API

Another popular use of API is to obtain the social graph of users' followers and friends, as well as a particular user's tweets. Below we show two common usage examples of the User APIs:

```
# Get a list of followers of a particular user
twitter.followers.ids(screen_name="cocoweixu")
```

```
# Get a particular user's timeline (up to 3,200 of his/her most recent tweets)
twitter.statuses.user_timeline(screen_name="billybob")
```

Rate Limit

Unlike Streaming API, **REST APIs have a strict rate limit** on how many requests you can send given a time limit and how many tweets you can get access to for each request (and it got stricter and stricter in the past). The limits vary from one API function to another. Twitter's dev website give a list of the rate limits here (https://dev.twitter.com/rest/public/rate-limits).

You can also query the API to check your remaining quota, though you may rarely use this command:

```
twitter.application.rate_limit_status()
```

6. Learning More about Twitter APIs

This tutorial is meant to help you to start. To learn more about Twitter APIs, here are two ways I found quite sufficient:

- Look up the documentation (https://dev.twitter.com/rest/public) of Twitter APIs to find the function you would like to use, then search in the source code (https://github.com/sixohsix/twitter/) of the Twitter Python Tools to see how to call it in your program.
- Check more example Python scripts that demonstrate interactions with the Twitter API via the Python Twitter Tools: https://github.com/ideoforms/python-twitter-examples (https://github.com/ideoforms/python-twitter-examples)