How to get Twitter Data from the Twitter REST and Streaming API

David M. Beskow

Carnegie Mellon University

29 January 2018

Collecting Data on the Web in General

1. What platform should I use?
2. Should I collect everything?
3. How much should I pay?
4. Is my collection method legal/ethical?
5. Do I need an IRB?
6. Can I share this data?
7. Real-time vs. Historical?
8. API vs. Scraping?

Why Twitter??

1. As an open micro-blog, Twitter often represents the macro conversation
2. Relatively broad penetration
3. Relatively easy to collect
4. Falls under creative commons license
5. Offers rich demographic, geographic, text, and network data

Streaming or REST API

1. Collect **historical** data with Rest API
   - Content Based
   - Geo-based (radius from point of interest)
   - User data

2. Collect **real-time** data with the Streaming API
   - Content based
   - Geo-oriented (bounding box)

What format is my data in?

1. JSON!
2. Related question, what the heck is JSON?
3. JSON is a simple format for sharing unstructured data

```
{
    'key': 'value',
    'user_name': 'network_science_guy',
    'tweet_text': 'Never teach code after lunch!'
}
```

Typically one JSON "object" per tweet/line of file

Twitter JSON Structure

1. Text
2. Coordinates
3. Created_at
4. favorite_count
5. favorited
6. id
7. Lang
8. ...

Full list of fields at:
`https://dev.twitter.com/overview/api/tweets`

Network

1. User x User
    - Mention
    - Following
    - Retweet

2. Hashtag Graphs
    - Co-occurrence
    - Bipartite graph: user x hash tag

3. Node attributes
    - Profile features: following count, creation date,...
    - Language patterns, geo coord., etc

How to do it

1. Option 1: Use some commercial data collecting services
2. Option 2: Get the ASU team to do it (TweetTracker)
3. Option 3: Do it yourself!
    - API credentials (https://apps.twitter.com/, show how. . .)
    - Find a programming language you're comfortable with
        - R - twitteR package
        - Python – tweepy is the most popular tool
        - Java – Hosebird is Twitter's own tool for connecting to the streaming API

Common Approaches

1. Track all tweets within the U.S. for 6 months
2. Follow 1000 users I think are interesting for 6 months, do a network analysis
3. Follow #brexit for 6 months, do a network analysis
4. ...

Common Practice #1

1. Hook in to the Streaming API with keywords and/or bounding box for a bit
2. Find users that are "interesting"
3. Use the Search API to collect all of these users' data
4. Try to get rid of bots, celebrities if I can help it

Pros: Relatively easy, fast
Cons: Results are limited to these streaming keywords/locations.
The resulting mentioning/retweeting networks are usually sparse.

Common Practice #2

1. Start with a set of seed users of interest
2. Collect timelines for these users
3. Find new users within one-step connection (mentioning, following, retweeting)
4. Repeat step 1.

Pros: Get comprehensive social links for a group of users.
Cons: Time consuming, relies on the choice of seed users.

DEMO

Install *twitter_col* Package

Before starting, you first must install the *twitter_col* package. The *twitter_col* is a package that I've written in Python to streamline common collection strategies.

```
pip install --user --upgrade
git+git://github.com/dmbeskow/twitter_col.git
```

Authenticate on Twitter API with Tweepy

```python
import tweepy

consumer_key ="xxxxxxxxxxxxxxxxxx"
consumer_secret = "xxxxxxxxxxxxxxxxxx"
access_token ="xxxxxxxxxxxxxxxxxx"
access_secret = "xxxxxxxxxxxxxxxxxx"

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_secret)

api = tweepy.API(auth, wait_on_rate_limit=True,
           wait_on_rate_limit_notify=True)
```

REST Scrape

Below is a basic script for a content related REST Scrape. This script will take a list of terms (no limit in number) and individually search twitter for any time these terms appear in the last week. It will save these into separate files by term. Note that each file will not contain duplicates, but that combining the files will create duplicates. The 'prefix' adds a string to the file names that allows you to differentiate separate projects (i.e. 'NBA' vs 'NFL' scrapes).

```
from twitter_col import scrape
terms = ['#NBA','#basketball','#jordan']

scrape.rest_scrape(api, searchQuery = terms,
                        prefix = 'NBA')
```

Setting up Virtualenv for Streaming

While all the rest of the normal functions are available even if you aren't in a virtual environment, the streaming command line interfaces will only work with a virtual environment.

We will create the virtual environment from the terminal in Mac or Linux or using the Windows Linux Subsystem (WSL) in Windows:

```
virtualenv -p python3 twitter-env
```

Then we activate the environment with the command

```
source twitter-env/bin/activate
```

Setting up Keys for Streaming

Both of the command line interfaces require the user to provide the path to a JSON file with their Twitter credentials. Having created your Twitter credentials, place them in a json file with the format below:

```
{
  "consumer_key": "XXXXXXXXXXXXXXXXXXXXXXXXXXXX",
  "consumer_secret": "XXXXXXXXXXXXXXXXXXXXXXXXXXXX",
   "access_token": "XXXXXXXXXXXXXXXXXXXXXXXXXXXX",
   "access_secret": "XXXXXXXXXXXXXXXXXXXXXXXXXXXX"
 }
```

Stream Based on Content

In this section I will introduce the *stream_content* command line interface (CLI) that facilitates easy access to the streaming API with content filtering. This allows you to filter the Streaming API by any token (hashtag, screen name, text, etc).

Let's say I want to stream content during the Worldcup in regards to Germany, France, and Spain. I could use their country hashtags with the CLI command:

```
stream_content key.json '#GER,#FRA,#ESP'
```

This CLI tool will create a new file every 20K tweets.

In this case, the resulting file will be named '#GER_#FRA_#ESP_YYMMDD-hhmmss.json.gz'. In general I find it is helpful to keep your search terms in the name of the file so you can remember how you obtained the data.

Stream Based on Geo

This allows you to filter the Streaming API by a rectangular bounding box (city, state, country, region). If you need to find bounding boxes for specific countries, I recommend https://gist.github.com/graydon/11198540.

Let's say we want to stream data for New York City. We could do this with the following command

```
stream_geo key.json -74 40 -73 41 -tag nyc
```

which produces a file named 'nyc.YYMMDD-hhmmss.json.gz'

Questions?