Twitter data collection

SCC.413: Applied Data Mining

Week 14, 6th February 2019

In this lab exercise you will interact with the Twitter REST API using Python code to download tweets for future analysis. Data collected via APIs are generally much cleaner than web scraped data, and also structured nicely (here, via JSON) for easy querying.

To collect data from Twitter you need to have a Twitter account, and also create an authorised application. If you do not want to do this, you could skip this lab and just use pre-collected data, although it is useful to see how to collect your own data. One option would be to partner with a neighbour and use a single Twitter account.

Ensure you have downloaded the code from the git repository (as described on Moodle), and place it in a folder for this lab. Your h-drive is probably the best place, although keep an eye on the space available with the various data files you will be creating in the lab.

The code provides a working tool for collecting Twitter data via the Twitter REST API. We use the `twython` Python package to assist us, although others are available, most notably `tweepy`.

You may need to install the `twython` package in Python on the VM, with:

```
$ pip install twython
```

1 Authorising the Twitter application

To get started, you will need to create an authorised Twitter application, follow these steps. You only need one Twitter application for the module, and you can use the same authorisation credentials for any future work on the module utilising the Twitter API.

1. Go to `https://apps.twitter.com`.
2. Sign into Twitter.

3. Click “Create an app”.
4. Unless you already have a developer account, you will be asked to create one, follow the instructions, selecting:
   (a) “I am requesting access for my own personal use”.
   (b) “Student project / Learning to code” for use case.
   (c) For the description of what you are building, fill in with your own words:
      i. Collecting tweets for analysis, learning natural language processing skills.
      ii. Analysing tweets and their textual content, using various natural language processing techniques, such as sentiment analysis and topic modelling.
      iii. No creation of tweets or content, just the collection of tweets.
      iv. Tweets will not be displayed to end users, only to be used for learning to use Twitter API.
   (d) “No” for being available to a government entity.
5. Once you have created a developer account and verified your email address, you will be able to “Create an app”
6. Fill in the name (e.g. “Lancaster scc-413”), a brief description, and a URL (e.g. `https://delta.lancs.ac.uk/barona/scc-413`). Do not enable sign in with Twitter, and leave the URLs and organization name blank. For the “Tell us how this app will be used”, fill in some details about how you will be collecting Tweets for analysis with natural language processing techniques as part of a University course on applied data mining.
7. Agree to the developer agreement (have a quick read), and create.
8. You can edit the Access Level to “Read Only” in the *Permissions* Tab.
9. In the *Keys and tokens* tab, click “Create” under “Access token & access token secret”.
10. You will need the *Consumer Key (API Key)*, *Consumer Secret (API Secret)*, *Access Token*, and *Access Token Secret*. Note, you should not share these as they are linked to your account with Twitter, and are rate-limited.
11. Copy and paste the generated authorisation details into the 4 relevant variables declared in `twitter_auth.py`.

2 Downloading a user’s tweets

The Twitter API allows for the downloading of any (unprotected) user’s tweets, limited to their last 3,200. Collecting a user’s tweets can be useful for various research questions, such as comparing language usage across individuals/organisations, and for performing various authorship analysis tasks (as we'll see later in the module).

You have been provided with Python code which will download the tweets of a provided Twitter user . To download the tweets of a user, run:

```
$ python3 user_tweets.py <username>
```

replacing `<username>` with a Twitter username (without the @), e.g. *LancasterUni*.

By default, a JSON file will be created, e.g. `LancasterUni_tweets.json` containing everything returned for each tweet from the Twitter API. This is a lot of information per tweet. The attributes are detailed at https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/tweet-object. Review the information that is available.

Review the code for `user_tweets.py`, and check your understanding of how it is collecting tweets. The Twitter API throttles the downloading of data, here allowing for 200 tweets per request, and 1,500 requests per 15-minute window. Therefore we need to collect 200 tweets at a time, with an older starting point each time, until there are no more tweets available. If we hit the rate limit, we pause the collection until the rate-limit window resets: https://developer.twitter.com/en/docs/basics/rate-limiting. Other options are available for the collecting the user tweets: https://developer.twitter.com/en/docs/tweets/timelines/api-reference/get-statuses-user_timeline.html. How would you discard tweets which are replies to other users?

3 Outputting tweets

`tweets_json.py` provides methods for outputting the tweets to a file for later use, and for loading saved tweets (JSON) for different outputs.

`to_full_json()` is used by default in `user_tweets.py` and `search_tweets.py`, outputting all information available from the API. You can change either to output a limited set of tweet attributes (`to_minimal_json()`, or just the plain text (`to_just_text()`).

Using the list of attributes available, try to produce a list of tweets (in JSON format or plain text, e.g. separated by a tab (TSV)) containing the time the tweet was written and the text. Of course, there are numerous ways the tweets could be outputted, e.g. into a database or a CSV file. Feel free to experiment with different outputs that might be useful to you.

4 Searching for tweets

The Twitter API also allows for the searching for Tweets, albeit only over a sample from the last 7 days (unless you pay). This could be useful for collecting a selection of tweets on specific topic, or mentioning people.

`search_tweets.py` provides capability for performing searches, in a similar manner to extracting a user's tweets (above). Review and check your understanding of the code. To download a collection of tweets matching a given search, run:

```
$ python3 search_tweets.py <search_string> <limit>
```

replacing `search_string` with a word, hashtag, or more complicated search, and `<limit>` with the maximum number of tweets to return, e.g. try:

```
$ python3 search_tweets.py "#lovelancaster" 500
```

to get 500 tweets using the *#lovelancaster* hashtag.

The are a number of search operators available, allowing for quite complex searches: https://developer.twitter.com/en/docs/tweets/search/guides/standard-operators. Ignore the instructions on URL encoding the search string, `twython` takes care of this for us.

There are also different parameters available for the search request itself: https://developer.twitter.com/en/docs/tweets/search/api-reference/get-search-tweets.html.

Come up with your own searches, and discuss with your neighbours. A few you can try:

1. Find tweets from the *@LancasterUni* account mentioning *snow*.
2. Find tweets mentioning *snow* and the *#lovelancaster* hashtag, which are not retweets.
3. Find tweets mentioning *snow* from the Lancaster area.
4. Find 10 positive tweets about snow, and 10 negative tweets about snow.

Also, think about what useful tweet attributes are available to output for the above searches.

5 Further/advanced tasks

The Twitter API provides access to further information about tweets, users, and plenty more. Here's a list of things you can try if you have time. Please feel free to make other suggestions.

1. Many other methods are available from `twython` linked to Twitter API requests: https://twython.readthedocs.io/en/latest/api.html. One potentially useful task you should be able perform by adapting the available code is to collect the user details of a given account (see `show_user`).

2. Expanding on 1., you could collect a list of users (e.g. a user's followers) and then collect all of their user information and tweets.

3. You can use the Twitter Streaming API to collect tweets in real-time as they are posted. See the instructions for implementing this with `twython`: https://twython.readthedocs.io/en/latest/usage/streaming_api.html, and attempt to collect all tweets mentioning a word of interest.