COSC 2671 Social Media and Network Analytics

Lab 3

Data Pre-Processing and Introductory Twitter Analysis

Learning outcomes:

- Learn/revise how to load and pre-process a social media instance (Twitter)
- Learn how to perform basic analytics on Twitter data using Python

Requirements:

- A PC with Internet connection and Python 3 installed

Resources:

- This lab worksheet
- Associated code in lab03Code.zip

Introduction

In this lab, we will aim to learn/revise about data loading and pre-processing. We will also perform some basic analytics to continue to get (re)familiar with Python and some of the available analytics libraries/tools. We use Twitter as the basis for this lab.

We first go through how to access Twitter's API then do some basic analyses.

Twitter Registration and Access Key

First, you'll need a Twitter account to use the API. If you don't have one, please register for one at https://twitter.com/home

Twitter offers several types of APIs to access its data. Recently, Twitter changed their API policies, and to register a developer account, which is needed to use their API requires more effort than in the past. Hence for this class, instead we have already applied for an organisational account (this course) and will add you to the account. Note if you have a developer account already, feel free to use that or can use the class one, we happy either way.

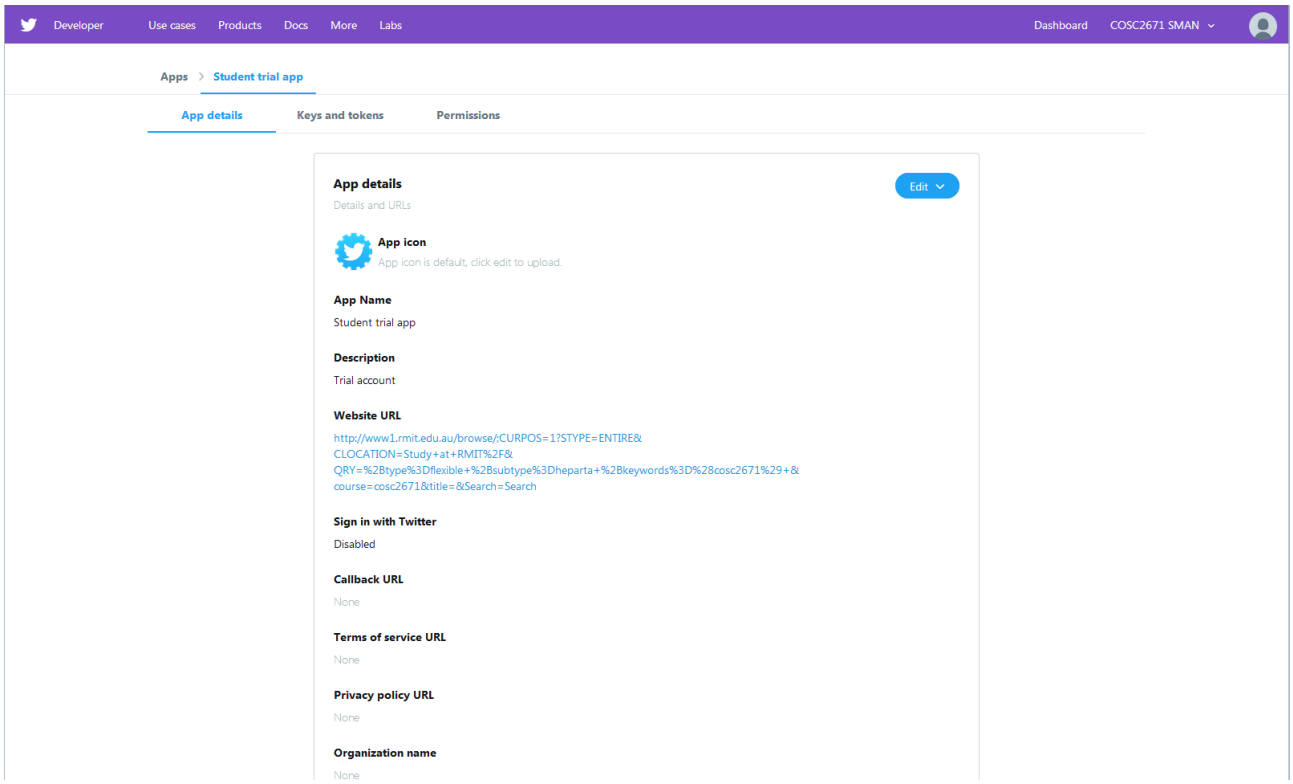
You'll need to enter your twitter handle at https://rmiteduau-my.sharepoint.com/:x:/g/personal/jeffrey_chan_rmit_edu_au/EVVu1S-izLBOp_tQynq2SP4BH8zH6lv13PirC5qB0l2p6A?e=wXaxRY . Let either Saiedur or myself know (in class or via email) then we can add your handle to the course Twitter account.

Once we added you to the developer account, using your favourite browser to go the Twitter Developer page https://developer.twitter.com/en.html (if you are not there already). Sign-in and click to create an app.

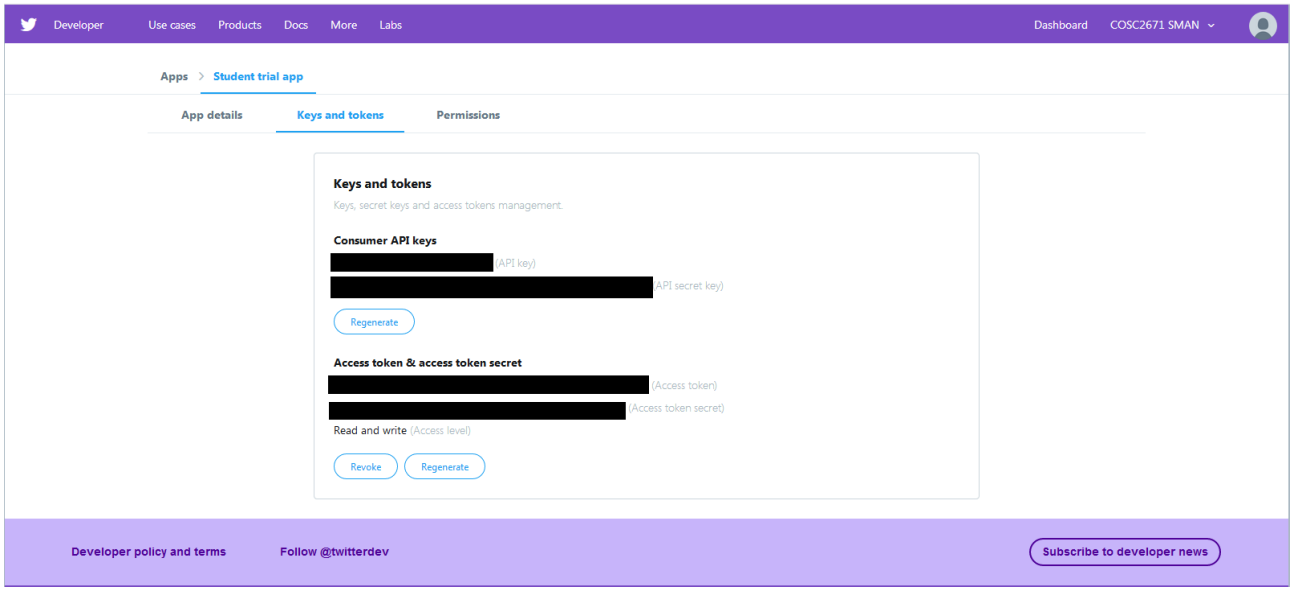
For the details of the app applicatoin, enter an appropriate app name (make it identifiable from everyone else's), add a very short description (e.g., it is an application for the course), disable sign

in from twitter, leave everything else blank apart from “app usage”, enter something along the lines that we are using this for educational purposes and doing Twitter analytics.

After creating the app, you’ll see something like the figure to below:



Under the *Keys and Access Tokens* tab at the top, there are several required keys and secret tokens. The *Consumer Key* and *Consumer Secret* (also known as *API Key* and *API Secret*) are for your application. The *Access Token* and *Access Token Secret* are for your user account (you might need to press “create” to generate these).



The Permissions tab and *Access Permission* defines what the application can do while interacting with Twitter on behalf of the user. Read-only is the most conservative option and the one we will be using for this laboratory. For this course, generally we don't to write as we are doing analytics and not generating Twitter content.

Twitter API

There are essentially two ways to access Twitter data via its APIs. The REST API allow us to search about existing published Tweets, but there are a number of restrictions. REST APIs only retrieve Tweets up to a certain length of time, and for the free accounts we are using, typically a week. In addition, it is a best effort retrieval, as in it is not guaranteed to return all Tweets over that length of time.

The other approach is to use the Streaming API, which essentially opens up a connection to Twitter and tweets that match our filter criteria are retrieved. We then retrieve and store as many Tweets as needed.

Both approaches have their uses. The Streaming API is the preferred approach if we wanted to collect a large volume of Tweets over a longer period of time, but obviously requires time and storage. The REST APIs are useful if we only wanted to search for Tweets authored by a specific user or our own timelines (these are all the Tweets we post, retweet etc).

Rate limit

Any discussion about collecting/accessing Tweets requires discussing about rate limits. Twitter APIs limits the amount of Tweets retrieved per time period – see https://dev.twitter.com/rest/public/rate-limiting for more details.

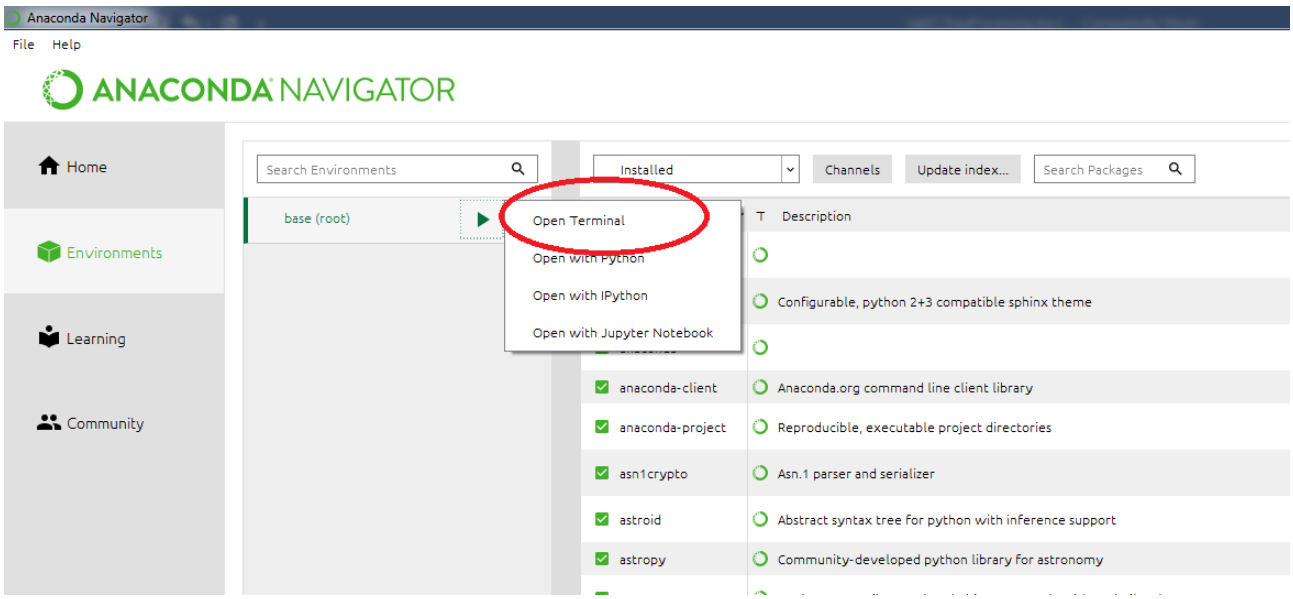
When your application or user has hit this limit, Twitter will return an error message. When this occurs, it is best to ease back the requests. If we continue to exceed the rate, your application and user will be blacklisted as potential abusers. One approach to do this is to sleep() for a number of seconds if we hit the rate limit. For this lab, it is unlikely we will hit the limit.

Next we are going to discuss the Python library we will use to help us access the APIs and how to setup this up to retrieve Tweets.

Python Twitter Client

There are several Python libraries/clients to connect to the Twitter API. The one we will use is Tweepy, as it is popular, has implemented many functionalities and is actively maintained.

The library is called *tweepy* if you are looking for it on Anaconda (or pip if you using that for library management). Note you may not be able to find tweepy in Anaconda Navigator, if not, open a terminal (see diagram below) and enter the conda installation commands from https://anaconda.org/conda-forge/tweepy



To connect to Twitter via tweepy, we will need the keys and secret tokens of the app you created earlier. Although it is generally not advisable to store these within plaintext Python scripts, for convenience for this lab, we will do just that. Alternatives is to store them as environment variables or use key management within the OS we are using.

If you haven't done so, download the associated code lab03Code.zip into your working directory. Unzip the file, and within the folder you'll find a file called *twitterClient.py*.
We have created some functions in twitterClient.py to connect to the Twitter APIs. Please open this file and examine its contents (you can use Jupyter notebook or your favourite Python editor). For this lab, you'll need to enter your keys and assess code into the the twitterAuth() function at the appropriate locations (see below for a code snippet, you replace with the appropriate keys and secret tokens where the … are).

```
try:
    consumerKey = ...
    consumerSecret = ...
    accessToken = ...
    accessSecret = ...
except KeyError:
```

The twitterAuth() function handles the authentication. The twitterClient() function creates a client Tweepy API client that will be used to connect and retrieve tweets. We will be using it for the rest of the lab.

Getting Tweets from the timeline

To get an understanding of the API, lets retrieve the first 10 tweets from your own timeline. Open a new Jupyter notebook and type the following into that script (one per line).

```
%load_ext autoreload
%autoreload 1
%aimport twitterClient
```

```
from tweepy import Cursor

client = twitterClient.twitterClient()

# retrieve the first 10 tweets in your timeline
for tweet in Cursor(client.home_timeline).items(10):
    # print out the tweet's text content
    print(tweet.text)
```

What do you see?

Next we want to modify the notebook to retrieve the timeline of specific users. Let's see what Tweets are associated with the RMIT CS account, @rmit_csit.

Exercise:
Modify the previous script to retrieve the timeline of @rmit_csit and print out the first 20 tweets.

Hint: A useful functionality in Tweepy is that it can retrieve a timeline of a user 'xyz' by changing the arguments to Cursor():

… Cursor(client.user_timeline, **screen_name='xyz'**).items(200)

Remember to replace 'xyz' with the right account name (hint, what account are we trying to observe?).

Streaming API

In this lab, although we will not directly use the Streaming API, the ideas are similar.

However, we will do a series of analysis for the rmit_csit timeline, which was retrieved earlier and stored in rmitCsTwitterTimeline.json (in json format).

Desirable

The following exercises are desirable and may be useful for your assignment 1. But you not expected to complete all of these within class (although that would be great). Come discuss during consultations if you have issues.

Frequent Hashtags

Looking at the top K number of hashtags associated with rmit_csit is useful to understand what the common associated topics are. Download *twitterHashtagFreq.ipynb*. Open the notebook and make sure you understand what every line is doing.

Run the notebook. What does it output?

Exercise:
Based on twitterHashtagFreq.ipynb, write a notebook that outputs the top 10 user mentions in the rmit_csit timeline. Examine tweepy's documentation to see how this can be done:

http://pythonhosted.org/tweepy/ and particularly https://gist.github.com/hrp/900964 (check-up user_mentions)

Text pre-processing

We analysed the hashtags and user mentions in the previous section. In this section, we extend our analysis to the unstructured part of Tweets, the text.

We will explore text processing and natural language processing in more detail later in the course, but for this lab we introduce some initial concepts.

There are a number of pre-processing steps to prepare the raw tweet text for processing. We will examine few of the main ones, including tokenisation, stop word removal and stemming.

Tokenisation converts a stream of text into a set of words (tokens). Many text mining and NLP approaches works on a set of words/tokens, hence this step is often the first pro-processing step.

Stop word removal is to remove frequent words that are common across many tweets and documents and not useful for text analysis. For example, stop words such as 'the' are very common and do not contain much additional information and hence should be removed, which is performed by this step.

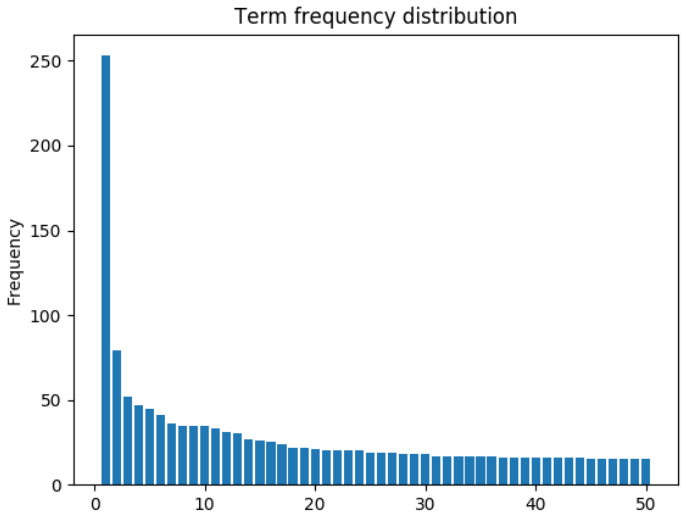
Finally stemming converts words that have different cases or singular and pural version of words into the same words.

We will be using the *nltk* library to do much of the heavy lifting for us. Nltk is an excellent library to do text pre-processing and NLP. It even comes with a tweet tokeniser (see TweetTokenizer() documentation in http://www.nltk.org/api/nltk.tokenize.html), that recognises hashtags and emoticons.

Download and open *tweetTextProcessing.ipynb* to see how tweets are pre-processed with tokenisation, stop word removal and normalisation. The script prints out the 50 most frequent words in the rmit_csit timeline.

Exercise:
Modify the previous script to filter out hash tags, emoticons and other noisy output, e.g., '...'.

Next, we are interested in producing a histogram of the term frequency distribution. This can be done, in a few lines, using matplotlib. Modify tweetTextProcessing.ipynb and use either the bar() or hist() function in matplotlib.pyplot to display the frequency distribution. You should be producing something similar to below:



What do you think the plot is telling us about the term frequencies? Do you think this is a common phenomenon in Twitter and other datasets?