Faculty of Computer Science, Dalhousie University *24/25-Oct-2023*

CSCI 4152/6509 — Natural Language Processing

Lab 7: Using Twitter API with Python

Lab Instructor: Sigma Jahan and Mayank Anand
Location: Goldberg CS 134(u)/CS 143(g)
Notes author: Colin Conrad, Dijana Kosmajac, Vlado Keselj

Using Twitter API with Python

Important Note: This lab cannot be completed with a free Twitter account and the current Twitter API. It is not required that students complete it. It is provided for information only.

Lab Overview

– Understand how Application Programming Interfaces (APIs) work
– Use the Tweepy package to extract tweets and profile information
– Clean text using Regex
– Export Tweets in a CSV format for later analysis

Step 1. Logging in to server `timberlea`

– Login to the server `timberlea`
  As in previous labs, you should login to your account on the server `timberlea`.
– Change directory to `csci4152` or `csci6509`
  Change your directory to `csci4152` or `csci6509`, whichever is your registered course. This directory should have been already created in your previous lab.
– Create the directory `lab7` and change your current directory to `lab7`:

```
mkdir lab7
cd lab7
```

Step 2: Register with Twitter API

So far, we have been using Python for data analysis which was already available to us through the NLTK package. We are now going to learn about using Twitter API to collect a dataset for analysis. API (Application Programming Interface), in a simple wording, is a software intermediary that allows two applications to talk to each other. APIs are often provided by platforms that want developers to build apps for them. For instance, Twitter provides an API, but so does Google Maps, Google Talk, Facebook, YouTube, AccuWeather, and many more. APIs let us bring outside computing power into our apps, and they do this using web services. Most web services use the REpresentational State Transfer (REST) protocol to provide the service. There are many resources on Internet where you can read about the REST protocol.

In order to access the REST API, we will need a Twitter developer account.

About Twitter Developer Account: As mentioned, this lab requires that you have a developer Twitter account. This has been a generally useful lab to show students how to collect their own data for a project, and how to use an API of a social media company. However, we want to avoid making any kind of requirements for you to open a

Twitter account and give out your private data if you do not want to, so we will first clarify the expectations and different options that you have.

First, an IMPORTANT NOTE: You are not required to open a Twitter account if you do not want to in order to get full marks for this lab. One option is just to write the code given in the lab and submit it, and also to submit a note explaining that you prefer not to access Twitter for the lab, and that will be sufficient for full marks.

The second option if you do not have a Twitter account is that you get in touch with the instructor by email, and we can send you access tokens that we generated, which you can use to access Twitter. Hopefully they will work, but since it is not coming from your own account they may not work.

The third option is that you use your Twitter account. This is the best option if you want to fully experiment with the lab and retrieve data from Twitter, but again, this is not required. Now, we will explain how to prepare your account and prepare the access tokens for later use. If you have a chance, it is a good idea to do this preparation ahead of the lab time, because you may need to wait for approval of your developer account. You should follow the following steps:

1. To open a Twitter account, you should go to `https://twitter.com` and sign up for an account. You do not have to use your real name, but you need to give your email address to verify developers account later. You will also be required to give your mobile phone number to open a developers account.

2. To open a developer account, you should, while logged in Twitter, go to `https://developer.twitter.com` and click on "Apply" to apply for a developer account. This will require that you have a verified mobile phone number associated with the account. In further steps, you can choose "Academic", "Student", and other appropriate parameters. You will not need to analyze or post data, or share with anyone, so you should not choose those options. Once you submit the application, it is verified via email.

3. You should create your project, and give an app name. For example, you can name your project and app NLPprj, choose use case "Student", give a project description as the lab in the course, and you will obtain the **API Key** and **API Key Secret,** which you should save in a text file. You should also save the Bearer Token which is generated. Then, you should go to "Keys and Tokens" and generate the Access Token and Secret. For this, you may need to got the Dashboard and click on an icon "keys and tokens". Save **Access Token** and **Access Token Secret** in you text file, and you should be ready for the lab.

If you want, you can keep open the last Twitter page so you can more easily copy the tokens later in the lab.

Step 3: Tweepy Package

Python has a package system which are essentially software libraries that extend basic Python functionality. For example, there is the `csv` package used to manage files in the CSV format, which is one of the basic libraries of Python and is maintained by the Python Software Foundation. Most packages however are maintained by a special community of dedicated users of that package. For example, we saw the package Natural Language Toolkit (`nltk`), then there is a library Tweepy (`tweepy`) for Twitter API access, Python Data Analysis Library (`pandas`) to better manage data for data analysis, SciPy (`scipy`) with additional numerical functionality, and others. To help manage these packages, the newer Python versions are shipped with a package manager called `pip`. You can read more about the Tweepy package at its web site at: `https://github.com/tweepy/tweepy`

Tweepy should be already available on the `timberlea` server. We will check if it is available by starting our a script, which we will call the Twitter-Profiler script. In the `lab7` directory, create a file named `lab7-twitter-profiler.py` and enter the following contents into it:

```
#!/local/bin/python
# File: lab7-twitter-profiler.py
# Twitter Profiler app. This is a simple script to configure the Twitter API

import tweepy     # Comment: https://github.com/tweepy/tweepy
import time
```

You can run this script using the command:

```
  python lab7-twitter-profiler.py
```

or, you can make the script user-executable by setting the appropriate permissions, and run it using the command:

```
  ./lab7-twitter-profiler.py
```

If you do not get any error, it means that the `tweepy` package is available; otherwise you would get an error as follows:

```
ModuleNotFoundError: No module named 'tweepy'
```

Installing `tweepy` as a root: The package `tweepy` is installed on `timberlea`, but if you are in a situation that you need a Python package that is not installed on a server, there are ways to install it. If you have root privileges, the most common way is to use those privileges to install it on the system by running a command like this:

```
  sudo pip install tweepy
```

The `sudo` command uses administrator (i.e., root) privileges, and the command `pip` is a command for installing Python packages.

Installing `tweepy` as a user: If you are working on a system with many users, like `timberlea`, chances are that you do not have root privileges, so you cannot install a package on the whole system. You can still install a Python package locally, to be used only by you, and in a certain application. This set-up is called a *local virtual environment* in Python. To install `tweepy` in a local virtual environment you would need to use the following commands:

```
  python -m venv .
  source ./bin/activate
  pip install tweepy
```

In this way, you could use the package `tweepy` even if it were not installed on the server as a whole.

Step 4: Create the Twitter-Profiler Script

Now, you can continue and complete the `lab7-twitter-profiler.py` script as follows:

```
#!/local/bin/python
# File: lab7-twitter-profiler.py
# Twitter Profiler app. This is a simple script to configure the Twitter API

import tweepy    # https://github.com/tweepy/tweepy
import time

# Twitter API credentials.  (You do not have to type the rest of this
# comment in your script.)
# The credentials below are just shown as an example.  If you created
# your own tokens you can use them here.  If you do not have Twitter
# account or tokens, you can check with instructor to obtain them.
# The third option is that you do not run the script and write a note
# why you could not use credentials in a file named: lab7-readme.txt

# If you have credentials, fill them below as strings as follows:
# consumer_key is the same as API Key
consumer_key = "FC1HCyMBei**********"
# consumer_secret is API Key Secret
consumer_secret = "lLXBeSYowLHHDCJS***********"
# access_key is Access Token
access_key = "2841901529-MAFxKe**********************"
# access_secret is Access Token Secret
access_secret = "6QcxzJR*********************"

# this function collects a twitter profile request and returns a
# Twitter object
def get_profile(screen_name):
    auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_key, access_secret)
    api = tweepy.API(auth)
    try:
        user_profile = api.get_user(screen_name=screen_name)
    except:
        user_profile = "broken"

    return user_profile

s = get_profile("DalhousieU")
print(s)
```

If you run this script, and have valid credentials, you should get output that would start like this:

```
User(_api=<tweepy.api.API object at 0x7fd49835d490>, _json={'id':
42391676, 'id_str': '42391676', 'name': 'Dalhousie University',
'screen_name': 'DalhousieU', 'location': 'Nova Scotia, Canada',
'profile_location': {'id': 'bb176f7b6355d404', 'url':
'https://api.twitter.com/1.1/geo/id/bb176f7b6355d404.json',
'place_type': 'unknown', 'name': 'Nova Scotia, Canada',
'full_name': 'Nova Scotia, Canada',
...
```

If you get simple output '`broken`', it means that probably credentials did not work, or there was some other error.

We will now explain the script a bit more: The import statements are used to import tweepy and time libraries. The time library makes it so that we can interpret time data structures, which is very helpful for interpreting tweets.

The `consumer_key` and other variables are the keys and tokens that are used by the Twitter API in order to be allowed to access Twitter and get data. We talked about how to get these keys and tokens from the Twitter developer interface.

We can take a look at the following function:

```
def get_profile(screen_name):
    auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_key, access_secret)
    api = tweepy.API(auth)
    try:
        user_profile = api.get_user(screen_name=screen_name)
    except:
        user_profile = "broken"

    return user_profile
```

The `auth` variable contains the OAuth object for logging into the Twitter API using your Twitter account credentials. The `auth.set_access_token` handles the other part of the Twitter OAuth process, which gets you into the API. Finally, the `api` variable saves the API session for use in the Python script. With this, we can use Tweepy and access Twitter's API securely. If you are interested about the OAuth standard used to access Twitter API, you can read more about in the Wikipedia page.

Step 5: Interpret Something Useful

We have code that works, but its output is very messy and not user readable. This is simply the Twitter API data. When you print Tweepy's Twitter API object, you will receive a lot of JSON, which is the format that the API uses to dump data.

Let's try modifying the script to only show some relevant features. Modify the end of the `lab7-twitter-profiler.py` to something like this:

```
s = get_profile("DalhousieU")
# print(s)
print("Name: " + s.name)
print("Location: " + s.location)
print("Description: " + s.description)
```

So, you the first '`get_profile`' is the old line, you comment out the '`print(s)`' line, and you add the three lines as shown. If you run the script again, you should get output similar to this:

```
Name: Dalhousie University
Location: Nova Scotia, Canada
Description: This is #DalhousieU - Atlantic Canada's leading
research-intensive university, just steps from the ocean.
```

That is a bit more usable. How did we change this? When the Twitter API creates a REST query, it makes a JSON request. JSON (JavaScript Object Notation) is widely adopted format for transferring web data. To get the raw JSON object directly, you can use `s._json` property, instead of accessing individual attributes.

The problem is that Python is not designed to interpret JSON. Tweepy is a library designed specifically to transform JSON data into a Python-readable Twitter object. This is why we are able to simply state `s.location` and receive the location. As you likely know, this is not normally allowed in Python.

Try tinkering with this. You can learn more about the Twitter profile object here: `https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/user-object`

You should create the file `lab7-readme.txt` by now. You can make a comment in it if you accessed Twitter API successfully, or if not, describe briefly the issue.

Submit: Submit the program `lab7-twitter-profiler.py` and the file `lab7-readme.txt` using the `submit-nlp` command.

Step 6: Retrieving a User's Tweets

So far, we have managed to get Python to print a given user's profile information. What about collecting user tweets? This is surprisingly simple because of how the Tweepy library structures the data. Recognizing that each user is a Twitter user object, it follows that a users' tweets must likewise be in some way related to that object. Profiles and tweets have what is often called a "one-to-many" relationship, where there are many tweets to a given user.

To search for ways to collect tweets, we can look at the Twitter API documentation. Take a quick look and try to see what is going on: `https://developer.twitter.com/en/docs/tweets/timelines/overview`

The docs say that we should use the `user_timeline` function to access a users' tweets. Immediately, we might think that the simplest way to do this is to call the `user_timeline` function on the user object. However, this will not work, as the `user_timeline` function is a call to the Twitter API object, not the user object. This is what is implied when the documentation requires a `screen_name` or id to make the call.

It makes more sense to create a separate `get_tweets` function that takes the `screen_name` as input and returns the user tweets. Let us try something like the following:

First, copy the previous `lab7-twitter-profiler.py` script into a new file called: `lab7-tweets.py` using command:

```
 cp -a lab7-twitter-profiler.py lab7-tweets.py
```

(We use the `-a` option with `cp` in order to preserve user-executable permission on the file.)
Now, you should edit the file `lab7-tweets.py` in the following way: First, replace the first part:

```
#!/local/bin/python
# File: lab7-twitter-profiler.py
# Twitter Profiler app. This is a simple script to configure the Twitter API

import tweepy    # https://github.com/tweepy/tweepy
import time
```

with the following:

```
#!/local/bin/python
# File: lab7-tweets.py
# Retrieving user tweets

import tweepy, time, csv
```

This is a simple description update, we import one more package `csv`, and show how to combine several packages in one `import` statement. After this we will leave everything that we had before, except deleting the following bottom part of the file:

```
s = get_profile("DalhousieU")
# print(s)
print("Name: " + s.name)
print("Location: " + s.location)
print("Description: " + s.description)
```

After deleting this part, we will first define a new function `get_tweets` for retrieving tweets of a user. It has a lot of similarity to the `get_profile` function, so you may want to start by copying `get_profile` function below itself, and by editing it. The function `get_tweets` should look as follows:

```
# this function collects twitter profile tweets and returns a Tweet
# object
def get_tweets(screen_name):
    auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_key, access_secret)
    api = tweepy.API(auth)
    try:
        # https://developer.twitter.com/en/docs/tweets/timelines/overview
        # describes user_timeline
        tweets = api.user_timeline(screen_name=screen_name, count=20)
    except:
        tweets = "broken"

    return tweets
```

After this function, add a couple line of code to test the script as follows:

```
t = get_tweets("DalhousieU")
print(t)
```

If you run the program with valid credentials, you should see the output in the JSON format, which looks as follows:

```
[Status(_api=<tweepy.api.API object at 0x7f3ce58be6d0>,
_json={'created_at': 'Mon Oct 25 21:00:14 +0000 2021',
'id': 1452741818211311619, 'id_str': '1452741818211311619', 'text':
'Here are 4 things you should experience this fall in Nova Scotia,
according to @DalStudentLife \U01F383 \U01F33D \u2615 \U01F342
https://t.co/OSEUqTC5Ty', 'truncated': False, 'entities':
{'hashtags': [], 'symbols': [], 'user_mentions':
[{'screen_name': 'DalStudent Life', 'name': 'Dal Student Life',
...
```

This result looks like a big mess of API Tweet results, but it is a good sign, because it means that our script is working. However, it needs to be further refined.

When using Tweepy, each tweet is an object in itself, complete with a series of properties which can be accessed. Our results currently give us a list of 20 Tweets, each with a lot of data. We can organize our data better by accessing just the text property of the tweets. Let us modify the code to loop through the data and give use the tweet content. Let us modify the code in the file `lab7-tweets.py` by commenting out the last print statement, and introducing a for-loop to print text of the tweets. The bottom of the file should now look like this:

```
t = get_tweets("DalhousieU")
#print(t)
for tweet in t:
    print(tweet._json["text"] + '\n')
```

You can run the program, and if you implemented this correctly and have valid credentials, the program should produce a short list of tweets which looks as follows:

```
Here are 4 things you should experience this fall in Nova Scotia,
according to @DalStudentLife \U01F383 \U01F33 \U01F33D \u2615
\U01F342 https://t.co/OSEUqTC5Ty

Interested in being #DalBound next year? Register for our November 6th
virtual Open House and: Explore student\u2026 https://t.co/jWwaicohHu

This week, learn about Amyl Ghanem, an associate professor in the
Department of Process Engineering and Applied  Sci

Our #DalUnited Way campaign starts today. Consider donating to this
worthy community cause. Our goal this year is t\u2026 https://t.co/g0yhnLaxbC

@mycampusgps Gorgeous photos!

\U01F642\U01F643 You won’t want to miss the ’Happiness in Troubled
Times’ virtual symposium, put on by @DAL_FASS. The event takes
...
```

Submit: Submit the program `lab7-tweets.py` using the `submit-nlp` command.

Step 7: Exporting Tweets to a CSV File

So far, we have managed to develop a script that collects a given user’s tweets. However, the program currently only prints the tweets that we find. We may want to save the tweets in a bit more structured format in a CSV file (comma-separated-values file which can be easily imported into a spreadsheet program like Excel).

We will write a script by making a copy of the previous script using the command:

```
  cp -a lab7-tweets.py lab7-tweets-csv.py
```

and modify the start of the new file as follows:

```
#!/local/bin/python
# File: lab7-tweets-csv.py
# Save tweets to a csv file

import tweepy, time, csv
```

We see that the script imports the `csv` library, which we did not use until now. Now, we will modify the bottom part of the script so that it does not simply prints tweets, but so that it saves each tweet into one row of a csv file. In addition to the text of a tweet, we might also want some other information, such as the id of the tweet, and the user who wrote it. We should also make sure to add a header for the csv file at the very beginning. The new end of a script should look something like this (just a draft):

```
with open (’lab7-tweets.csv’, ’w’) as outfile:
    writer = csv.writer(outfile)
    writer.writerow([]) # will insert feature names here
    for tweet in t:
        writer.writerow([])  # write feature values here
```

You should now got the end of the script and delete the following part:

```
t = get_tweets("DalhousieU")
#print(t)
for tweet in t:
    print(tweet._json["text"] + '\n')
```

and replace it with the following code:

```
# we list here some interesting Twitter profiles whose tweets we want to collect
profiles = ["DalhousieU", "google", "msdev", "CBCNS"]

# we open csv file for writing
with open('lab7-tweets.csv', 'w') as outfile:
    writer = csv.writer(outfile)
    # write header row with interesting features
    writer.writerow(["id","screen_name","created_at","text"])
    for profile in profiles:
        t = get_tweets(profile)
        for tweet in t:
            writer.writerow([tweet.id,tweet.user.screen_name,
                            tweet.created_at,tweet.text.encode("utf-8")])
            # another way to write this row is
            # tweet = tweet._json
            # writer.writerow([tweet["id"],tweet["user"]["screen_name"],
            #   tweet["created_at"],tweet["text"].encode("utf-8")])
```

If you implemented this correctly and have valid credentials, you can run the script and it should create a `lab7-tweets.csv` file which would look as follows:

```
id,screen_name,created_at,text
1452741818211311619,DalhousieU,2021-10-25 21:00:14+00:00,"b'Here are 4 things...
1452720376467922948,DalhousieU,2021-10-25 19:35:02+00:00,b'Interested in bein...
1452666499047559180,DalhousieU,2021-10-25 16:00:57+00:00,"b'This week, learn ...
...
```

The CSV file should include 20 latest tweets from each of the profiles: Dalhousie University, Google, Microsoft, and CBC Nova Scotia news. You can open it also in Excel format for better viewing. One issue with Excel is that the first column will be converted to numbers instead of treating them as strings.

Submit: Submit the program `lab7-tweets-csv.py` and the file `lab7-tweets.csv` using the `submit-nlp` command, or by uploading at the course web site. If you do not have Twitter credentials and cannot create the `lab7-tweets.csv` file, you can download it from the course web site under the 'Misc' label, or you can use this short URL: `https://vlado.ca/nlp/misc` using the common userid and password communicated on the class email list.

Step 8: Processing with NLTK

We have saved the CSV file with tweets collected from Twitter, or you can download it from the above site. Now, we will look at a short exercise to extract some information from collected data. You should start a new Python script called `lab7-hashtags.py` and start it as follows:

```
#!/local/bin/python
# File: lab7-hashtags.py
# Find tweet hashtags and tokens

import csv, nltk

with open ('lab7-tweets.csv', 'r') as infile:
    reader = csv.reader(infile,quotechar='"')
    for row in reader:
        print(row)
```

You can run the script and you should see the output like this:

```
['id', 'screen_name', 'created_at', 'text']
['1452741818211311619', 'DalhousieU', '2021-10-25 21:00:14+00:00',
"b'Here are 4 things you should experience this fall in Nova Scotia,
according to @DalStudentLife \\xf0\\x9f\\x8e\\x83 \\xf0\\x9f\\x8c
...
```

This is just an initial test. You should continue writing the `lab7-hashtags.py` file by replacing the `print(row)` command with some other commands to analyze tokens. Use the NLTK library to extract word tokens from each tweet (no tweet cleaning necessary). Print to the standard output unique word tokens per account. Then from each token list per account extract hashtags and print them as well. The output should look like this:

```
DalhousieU:
* unique tokens: Here, are, things, ...
* hashtags: #DalBound, #DalUnited, ...
Google:
* unique tokens: Hi, Shanaya, Without, ...
* hashtags:
msdev:
...
```

The order of the tokens and hashtags is not important, but they must print all of them and they should not be repeated. you can notice that in the sample of Google tweets we did not find any hashtags, so we did not print any.

Submit: Submit the program `lab7-hashtags.py` using the `submit-nlp` command, or by uploading at the course web site.

This is the end of Lab 7.