Michelle Graham
Assignment 2
CS 432
Spring 2017

P1. In order to extract 1,000 unique links from Tweets, I used the template found at:

http://adilmoujahid.com/posts/2014/07/twitter-analytics/

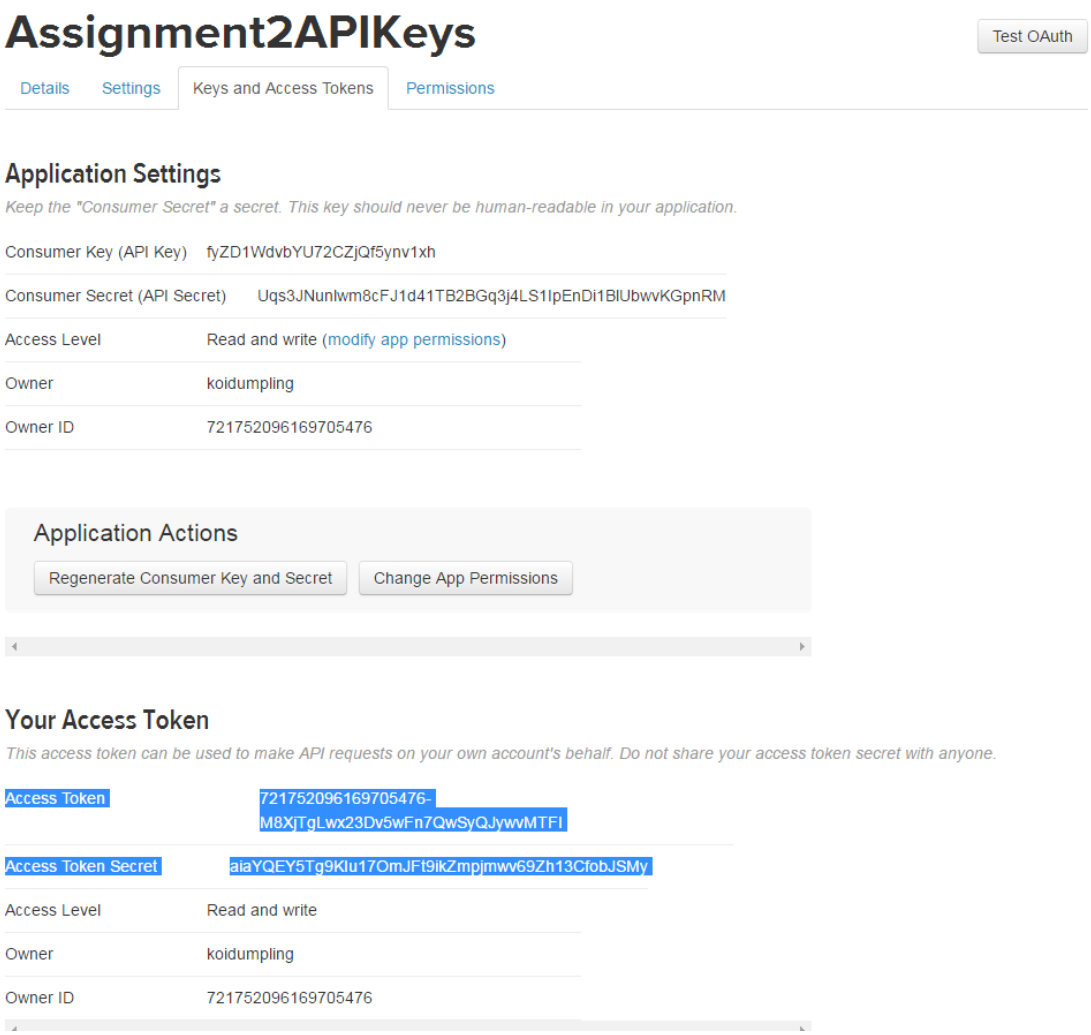
Template:

```
#Import the necessary methods from tweepy library
from tweepy.streaming import StreamListener
from tweepy import OAuthHandler
from tweepy import Stream

#Variables that contains the user credentials to access Twitter API
access_token = "ENTER YOUR ACCESS TOKEN"
access_token_secret = "ENTER YOUR ACCESS TOKEN SECRET"
consumer_key = "ENTER YOUR API KEY"
consumer_secret = "ENTER YOUR API SECRET"


#This is a basic listener that just prints received tweets to stdout.
class StdOutListener(StreamListener):

    def on_data(self, data):
        print data
        return True

    def on_error(self, status):
        print status


if __name__ == '__main__':

    #This handles Twitter authetification and the connection to Twitter Streaming API
    l = StdOutListener()
    auth = OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_token_secret)
    stream = Stream(auth, l)

    #This line filter Twitter Streams to capture data by the keywords: 'python', 'javascript', 'ruby'
    stream.filter(track=['python', 'javascript', 'ruby'])
```

Using this template, I created a python file to handle the code. In addition to this, I made an app on Twitter so that I could gain access to the API.

Michelle Graham
Assignment 2
CS 432
Spring 2017

Then, I replaced the access token, access token secret, consumer key, and consumer key secret with my own authentication information.



After doing this, I ran into an issue of trying to figure out what exactly “data” was referring to. It returned a bunch of unwanted data and I wanted to limit it to just the Tweet itself, keywords/ hashtags, users, and the unique urls. I had a hard time figuring this out. Luckily, I was able to find another example with a data limiter that also made it more readable. For every url that is extracted from the collected data, a unique, extended uri is supposed to be displayed. I tried to make this pretty by adding a couple of line spaces between each hit.

Michelle Graham
Assignment 2
CS 432
Spring 2017

A2CS432 - [C:\Users\mgraha\PycharmProjects\A2CS432] - Z:\CS432\A2\P1\twitter_streaming.py - PyCharm Community Edition 2016.3.1

File Edit View Navigate Code Refactor Run Tools VCS Window Help

Z: CS432 A2 P1 twitter_streaming.py twitter_streaming (1)

twitter_streaming.py

```python
# Import the necessary methods from different libraries
import tweepy
from tweepy.streaming import StreamListener
from tweepy import OAuthHandler
from tweepy import Stream
import json

# Variables that contains the user credentials to access Twitter API
access_token = "721752096169705476-M8XjTgLwx23Dv5wFn7QwSyQJywvMIFI"
access_token_secret = "aiaYQEY5Tg9KIu17OmJFt9ikZmpjmwy69Zh13CfobJSMy"
consumer_key = "fyZD1WdvbYU72CZjQf5ynv1xh"
consumer_secret = "Uqs3JNunlwm8cFJ1d41TB2BGq3j4LS1IpEnDi1B1UbwvKGpnRM"

# Accessing tweepy API
# api = tweepy.API(auth)

# This is a basic listener that just prints received tweets to stdout.
class StdOutListener(StreamListener):
    def on_data(self, data):
        # resource: http://code.runnable.com/Us9rrMiTWf9bAAW3/how-to-stream-data-from-twitter-with-tweepy-for-python
        # Twitter returns data in JSON format - we need to decode it first
        decoded = json.loads(data)

        # resource: http://socialmedia-class.org/twittertutorial.html
        # 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.
        count = 1000

        for url in decoded["entities"]["urls"]:
            count -= 1
            print "%s" % url["expanded_url"] + "\r\n\n"
            if count <= 0:
                break

    def on_error(self, status):
        print status


if __name__ == '__main__':
    # This handles Twitter authetification and the connection to Twitter Streaming API
    l = StdOutListener()
    auth = OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_token_secret)
    stream = Stream(auth, l)

    # This line filter Twitter Streams to capture data by the keyword: YouTube
    stream.filter(track=['YouTube'])
```

30:9 CRLF UTF-8

I attempted to create a counter so that the program would execute after 1,000 iterations. However, I am not sure this is properly implemented, as my code tends to break fairy quickly. Everything else is identical to the initial template.

Resource: http://socialmedia-class.org/twittertutorial.html

Michelle Graham
Assignment 2
CS 432
Spring 2017

```
Run twitter_streaming (1)

http://youtu.be/6_tGIt5MrtA?aDESAFIO

https://twitter.com/kam20013/status/832423748196241408

http://youtu.be/IYonPbptopE

http://youtu.be/IYonPbptopE

http://youtu.be/KG56B924PC8?a

144:1
```

This works pretty well aside from the random key error I get occasionally:

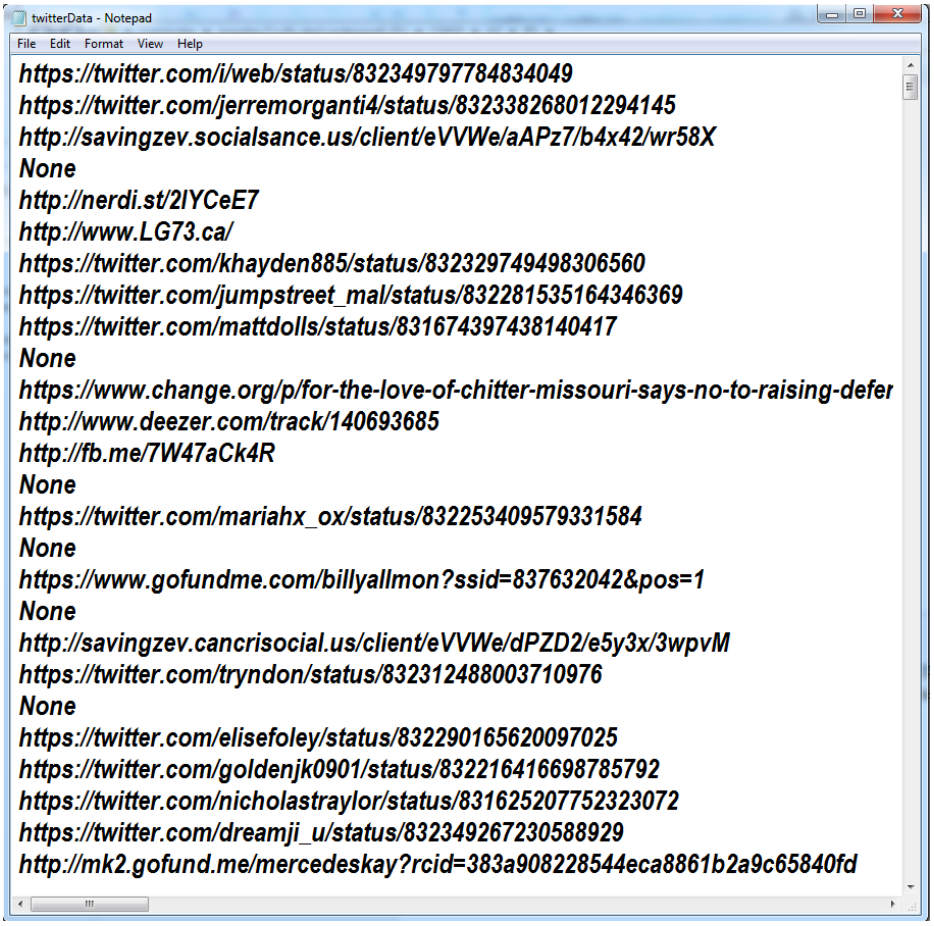
```
Run twitter_streaming (1)

Traceback (most recent call last):
  File "Z:/CS432/A2/P1/twitter_streaming.py", line 49, in <module>
    stream.filter(track=['YouTube'])
  File "build\bdist.win-amd64\egg\tweepy\streaming.py", line 450, in filter
  File "build\bdist.win-amd64\egg\tweepy\streaming.py", line 364, in _start
  File "build\bdist.win-amd64\egg\tweepy\streaming.py", line 297, in _run
  File "build\bdist.win-amd64\egg\tweepy\streaming.py", line 266, in _run
  File "build\bdist.win-amd64\egg\tweepy\streaming.py", line 327, in _read_loop
  File "build\bdist.win-amd64\egg\tweepy\streaming.py", line 300, in _data
  File "Z:/CS432/A2/P1/twitter_streaming.py", line 31, in on_data
    for url in decoded["entities"]["urls"]:
KeyError: 'entities'

Process finished with exit code 1
```

I used PuTTy to dump the links into a text file by using this command:

python twitter_streaming.py > twitterData.txt

Michelle Graham
Assignment 2
CS 432
Spring 2017

One Result:

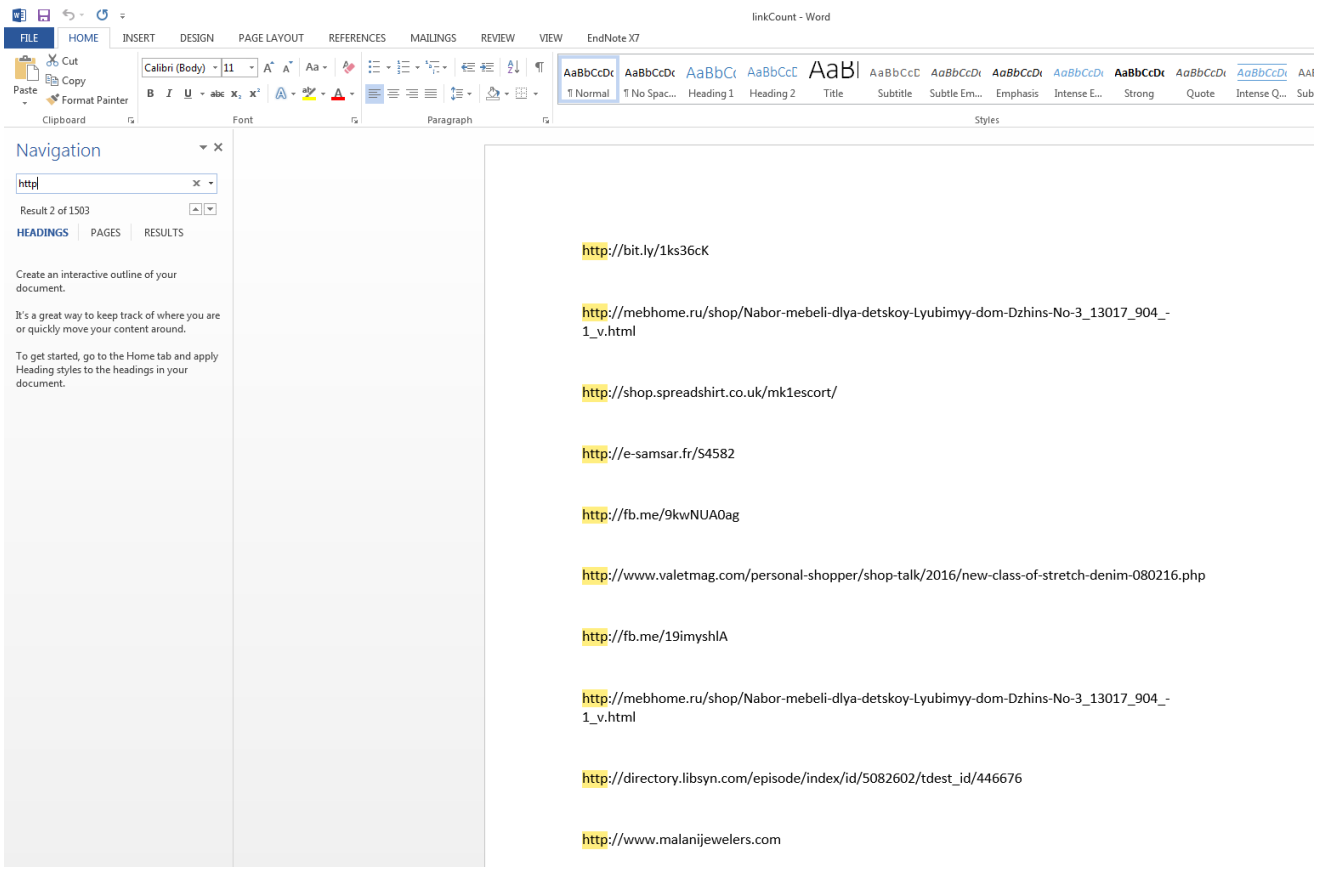


In order to get rid of the twitter links and missing urls and count the resulting URIs, I opened the text file that I dumped the data into with Microsoft Word- the layout is prettier and it enables the use of a search function to facilitate the counting. It also made it easier to get rid of the undesired links.

Because I figured this out so late, I used a bunch of different search terms that I thought might be popular like: YouTube, baby, health, money, donate, etc… I was trying to get as many unique links as possible in the shortest amount of time. This was a very interesting experiment- looking at how popular different search terms are!

Michelle Graham
Assignment 2
CS 432
Spring 2017

Sample Result:



I checked the validity of some of the links manually, but I did not do so exhaustively.

P2.