

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 authentication 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.

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

Assignment2APIKeys

Test OAuth

[Details](#) [Settings](#) [Keys and Access Tokens](#) [Permissions](#)

Application Settings

Keep the "Consumer Secret" a secret. This key should never be human-readable in your application.

Consumer Key (API Key)	fyZD1WdvbYU72CZJQf5ynv1xh
Consumer Secret (API Secret)	Uqs3JNunlw8cFJ1d41TB2BGq3j4LS1IpEnDi1BIUbwwKGpnRM
Access Level	Read and write (modify app permissions)
Owner	koidumpling
Owner ID	721752096169705476

Application Actions

Regenerate Consumer Key and Secret

Change App Permissions

Your Access Token

This access token can be used to make API requests on your own account's behalf. Do not share your access token secret with anyone.

Access Token	721752096169705476-M8XjTgLwx23Dv5wFn7QwSyQjywwMTFI
Access Token Secret	alaYQEY5Tg9Klu17OmJFt9IkZmpjmwv69Zh13CfobJSMY
Access Level	Read and write
Owner	koidumpling
Owner ID	721752096169705476

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.

```
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 x
1 # Import the necessary methods from different libraries
2 import tweepy
3 from tweepy.streaming import StreamListener
4 from tweepy import OAuthHandler
5 from tweepy import Stream
6 import json
7
8 # Variables that contains the user credentials to access Twitter API
9 access_token = "721752096169705476-M8XjTgLwx23Dv5wFn7QwSyQJywyMTFI"
10 access_token_secret = "aiaYQEX5Ig9KIu17OmJFt9ikZmpjmxv69Zh13CfobJSMY"
11 consumer_key = "fyZD1WdvbYU72CZjQf5ynv1xh"
12 consumer_secret = "Uqs3JNunlw8cFJ1d41TB2BGq3j4LS1IpEnD11B1UpwvKGpnRM"
13
14 # Accessing Tweepy API
15 # api = tweepy.API(auth)
16
17 # This is a basic listener that just prints received tweets to stdout.
18 class StdOutListener(StreamListener):
19     def on_data(self, data):
20         # resource: http://code.runnable.com/Us9rrMiTWf9bAAW3/how-to-stream-data-from-twitter-with-tweepy-for-python
21         # Twitter returns data in JSON format - we need to decode it first
22         decoded = json.loads(data)
23
24         # resource: http://socialmedia-class.org/twittertutorial.html
25         # Print each tweet in the stream to the screen
26         # Here we set it to stop after getting 1000 tweets.
27         # You don't have to set it to stop, but can continue running
28         # the Twitter API to collect data for days or even longer.
29         count = 1000
30
31         for url in decoded["entities"]["urls"]:
32             count -= 1
33             print "%s" % url["expanded_url"] + "\r\n\r\n"
34             if count <= 0:
35                 break
36
37     def on_error(self, status):
38         print status
39
40
41 if __name__ == '__main__':
42     # This handles Twitter authentication and the connection to Twitter Streaming API
43     l = StdOutListener()
44     auth = OAuthHandler(consumer_key, consumer_secret)
45     auth.set_access_token(access_token, access_token_secret)
46     stream = Stream(auth, l)
47
48     # This line filter Twitter Streams to capture data by the keyword: YouTube
49     stream.filter(track=['YouTube'])
50
```

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 fairly quickly. Everything else is identical to the initial template.

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



```
Run twitter_streaming (1)

http://youtu.be/6 tGIt5MrtA?aDESAFIO

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

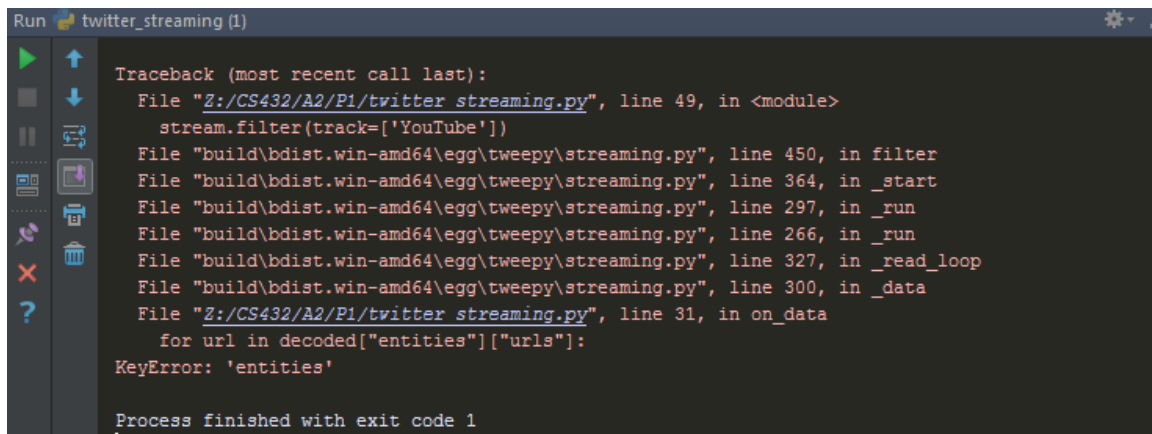
http://youtu.be/IYonPbptopE

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)

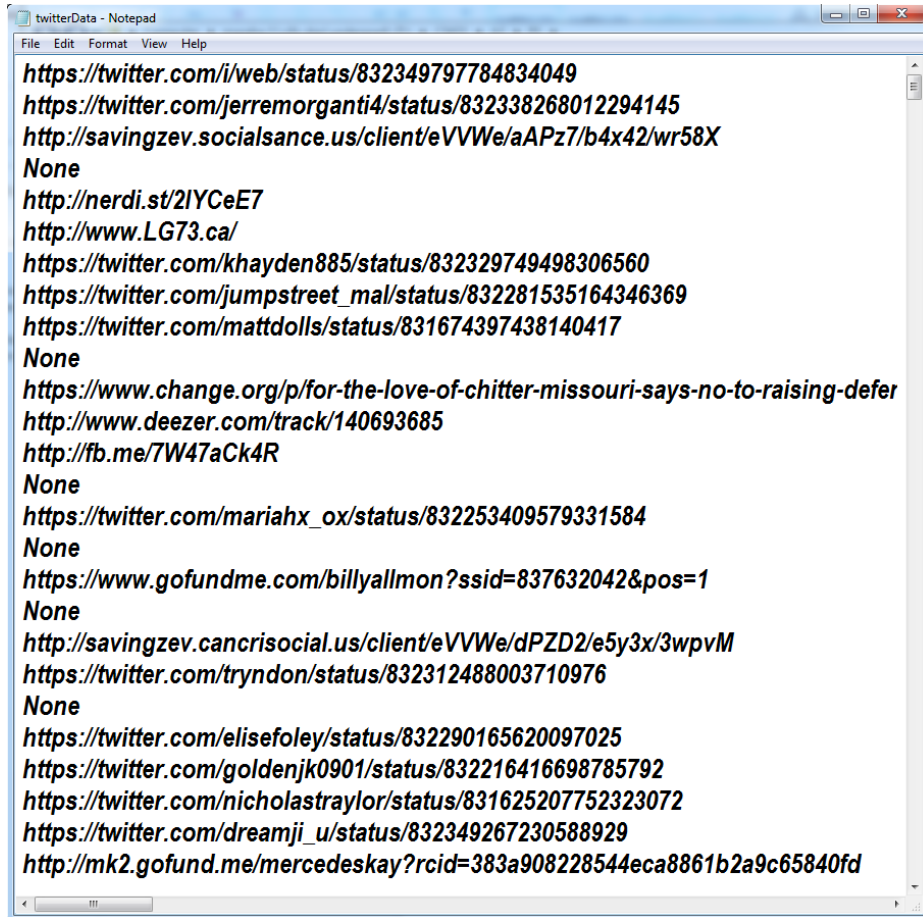
Traceback (most recent call last):
  File "Z:/CS432/A2/P1/twitter_streaming.py", line 49, in <module>
    stream.filter(track=['YouTube'])
  File "build\bdist.win-amd64\egg\tweepy\streaming.py", line 450, in filter
  File "build\bdist.win-amd64\egg\tweepy\streaming.py", line 364, in _start
  File "build\bdist.win-amd64\egg\tweepy\streaming.py", 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
```

One Result:

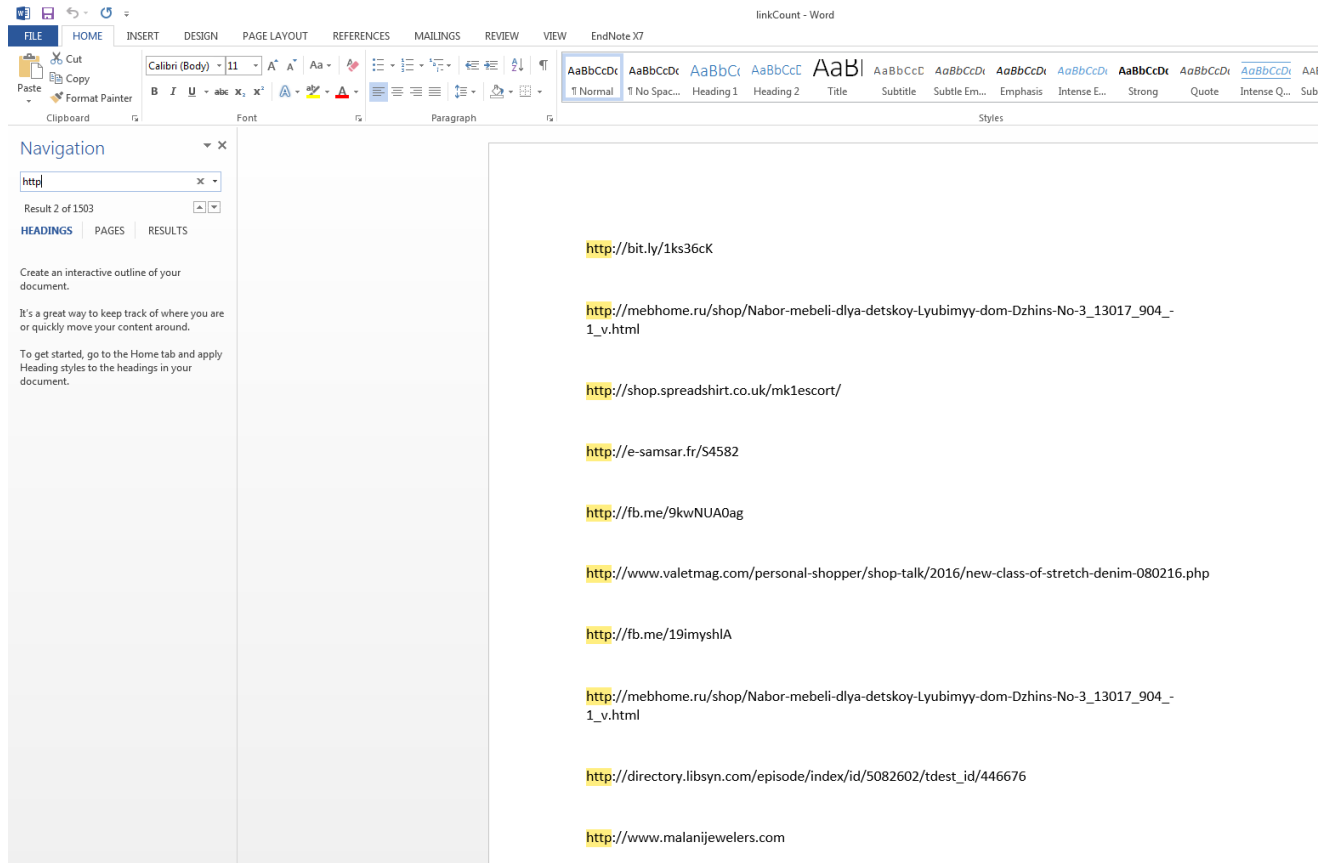


```
File Edit Format View Help
https://twitter.com/i/web/status/832349797784834049
https://twitter.com/jerremorganti4/status/832338268012294145
http://savingzev.socialsance.us/client/eVVWe/aAPz7/b4x42/wr58X
None
http://nerdi.st/2IYCeE7
http://www.LG73.ca/
https://twitter.com/khayden885/status/832329749498306560
https://twitter.com/jumpstreet_mal/status/832281535164346369
https://twitter.com/mattdolls/status/831674397438140417
None
https://www.change.org/p/for-the-love-of-chitter-missouri-says-no-to-raising-defer
http://www.deezer.com/track/140693685
http://fb.me/7W47aCk4R
None
https://twitter.com/mariah_x/status/832253409579331584
None
https://www.gofundme.com/billyallmon?ssid=837632042&pos=1
None
http://savingzev.cancrisocial.us/client/eVVWe/dPZD2/e5y3x/3wpvM
https://twitter.com/tryndon/status/832312488003710976
None
https://twitter.com/elisefoley/status/832290165620097025
https://twitter.com/goldenjk0901/status/832216416698785792
https://twitter.com/nicholastraylor/status/831625207752323072
https://twitter.com/dreamji_u/status/832349267230588929
http://mk2.gofund.me/mercedeskay?rcid=383a908228544eca8861b2a9c65840fd
```

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!

Sample Result:



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

P2.