Web Science: Assignment #2

Alexander Nwala

Mohd. Nauman Siddique

Saturday, February 16, 2019

Contents

Problem 1

Write a Python program that extracts 1000 unique (collect more e.g., 1300 just in case) links from Twitter. Omit links from the Twitter domain (twitter.com).
You might want to take a look at:

1. https://pythonprogramming.net/twitter-api-streaming-tweets-python-tutorial/
2. http://adilmoujahid.com/posts/2014/07/twitter-analytics/

see also:

1. http://docs.tweepy.org/en/3.7.0/getting_started.html#introduction
2. https://dev.twitter.com/rest/public

But there are many other similar resources available on the web. Note that only Twitter API 1.1 is currently available; version 1 code will no longer work.
Also note that you need to verify that the final target URI (i.e., the one that responds with a 200) is unique. You could have many different shortened URIs for www.cnn.com (t.co, bit.ly, goo.gl, etc.). For example:

```
$ curl -IL --silent https://t.co/DpO767Md1v | egrep -i "(HTTP/1.1|^location:)"
HTTP/1.1 301 Moved Permanently
location: https://goo.gl/40yQo2
HTTP/1.1 301 Moved Permanently
Location: https://soundcloud.com/roanoketimes/ep-95-talking-hokies-recruiting-one-week
    -before-signing-day
HTTP/1.1 200 OK
```

You might want to use the streaming or search feature to find URIs. If you find something inappropriate for any reason you see fit, just discard it and get some more links. We just want 1000 links that were shared via Twitter.
Hold on to this collection and upload it to github – we'll use it later throughout the semester.

SOLUTION

The solution to the problem is as below:

1. **Collect Tweets**: *Python-Twitter*, a Python-wrapper library over Twitter API has been used to collect tweets. *GetStreamFilter* API allows us to get stream of tweets by setting the track argument to the expressions to search for. I used trending hastags to collect stream of tweets. In total, I collected 211901 tweets.

```
'''
Fetch Twitter Stream
'''


def fetch_twitter_stream():
    file_write = open("StreamOutput2.txt", "w")
    api = create_twitter_instance()
    stream_response = api.GetStreamFilter(track="Jussie, #HTGAWM, Nadia")
    for i in stream_response:
        file_write.write(str(i) + "\n")
    file_write.close()
```

Problem 1 continued on next page...

2. **Extracting Links**: From the extracted tweets, I parsed out the urls from the tweet text and stored them onto another file. The urls contained the t.co shortened urls. The shortened urls where chased down for all the redirects and removing all the twitter.com domain urls and stored in a file. I did not use all the tweets in collecting the urls. I parsed 2088 urls.

```
'''
Parse Stream Output
'''


def parse_stream_output():
    file_open = open("StreamOutput1.txt", "r")
    file_write = open("Urls.txt", "w")
    for line in file_open:
        tweet_output = ast.literal_eval(line)
        print(tweet_output)
        if "limit" in tweet_output:
            continue
        tweet_text = (tweet_output["text"])
        tweet_url = parse_url_from_text(tweet_text)
        if tweet_url is not None:
            file_write.write(tweet_url + "\n")
    file_write.close()
    file_open.close()


'''
Check Urls fetched from tweets
'''


def check_urls():
    file_open = open("Urls_Uniq.txt", "r")
    file_urls_expand = open("Urls_Uniq_Expanded.txt", "a+")
    for line in file_open:
        print(line)
        try:
            url = line.rstrip()
            count = 0
            while True:
                if count > 10:
                    break
                response = requests.head(url)
                print(response.status_code)
                if 300 < response.status_code < 400:
                    url = response.headers['location']
                    count += 1
                else:
                    file_urls_expand.write(url + "|||" + str(response.status_code)
                        + "\n")
                    break
        except Exception as e:
            print(e)
            print(line.rstrip())
```

```
    file_open.close()
    file_urls_expand.close()


'''
Parse Url from a Text
'''


def parse_url_from_text(text_string):
    tweet_url = re.search("(?P<url>https?://[^\s]+)", text_string)
    if tweet_url is not None:
        tweet_url = tweet_url.group("url")
    return tweet_url
```

Problem 2

Download the TimeMaps for each of the target URIs. We'll use the ODU Memento Aggregator, so for example:
URI-R = http://www.cs.odu.edu/
URI-T = http://memgator.cs.odu.edu/timemap/link/http://www.cs.odu.edu/
or:
URI-T = http://memgator.cs.odu.edu/timemap/json/http://www.cs.odu.edu/
(depending on which format you'd prefer to parse)
Create a histogram* of URIs vs. number of Mementos (as computed from the TimeMaps). For example, 100 URIs with 0 Mementos, 300 URIs with 1 Memento, 400 URIs with 2 Mementos, etc. The x-axis will have the number of mementos, and the y-axis will have the frequency of occurence.
* = https://en.wikipedia.org/wiki/Histogram
What's a TimeMap?
See: http://www.mementoweb.org/guide/quick-intro/
And the week 4 lecture.

SOLUTION

```
'''
Function to count timemaps
'''


def fetch_timemaps():

    file_urls_expand = open("Urls_Uniq_Expanded.txt", "r")
    file_timemaps = open("Urls_timemap.txt", "w")
    for line in file_urls_expand:
        try:
            command = "http://localhost:1208/timemap/json/"
            command = command + line.rstrip().split("|||")[0]
            response = requests.head(command)
```

```
            file_timemaps.write(line.rstrip().split("|||")[0] + "|||" + str(response.
                headers["X-Memento-Count"]) + "\n")

        except Exception as e:
            exit()
    file_timemaps.close()
    file_urls_expand.close()
```

The solution for this problem is outlined by the following steps:

Find memento Count: I did not fetch the timemaps to count the memento count, but rather I made a head request using memgator and relied on its *X-Memento-Count* response header to know the number of mementos. I was able to find mementos for 1696 urls.
Histogram: The histogram shows a skewed graph with 1691 urls having less than 20,000 mementos. 3 urls have 20,000 to 40,000 mementos and 2 urls have 60,000 to 80,000 mementos.

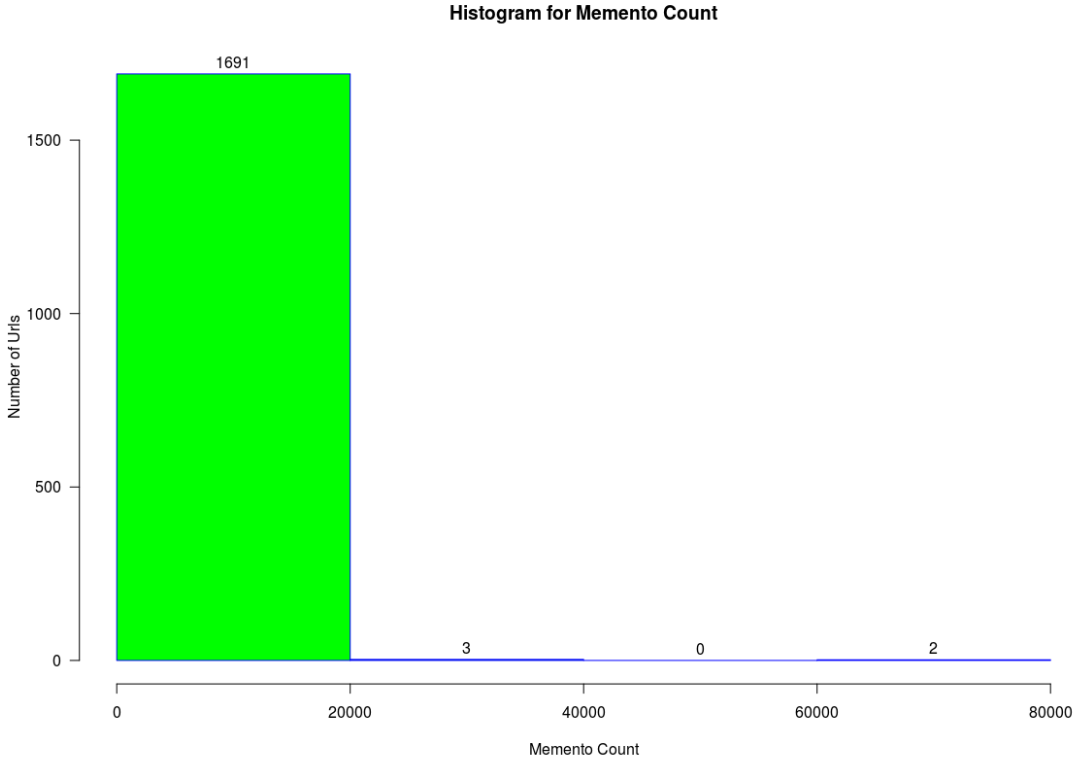

Figure 1: Histogram for Memento Distribution

Problem 3

Estimate the age of each of the 1000 URIs using the "Carbon Date" tool:
http://ws-dl.blogspot.com/2017/09/2017-09-19-carbon-dating-web-version-40.html
Note: you should use "docker" and install it locally. You can do it like this:
http://cd.cs.odu.edu/cd?url=http://www.cs.odu.edu/
But it will inevitably crash when everyone tries to use it at the last minute.

For URIs that have ¿ 0 Mementos and an estimated creation date, create a graph with age (in days) on the x-axis and number of mementos on the y-axis.
Not all URIs will have Mementos, and not all URIs will have an estimated creation date. Show how many fall into either categories. For example,
total URIs: 1000
no mementos: 137
no date estimate: 212

SOLUTION

```
'''
Function to carbondate urls
'''


def carbondate_urls():

    file_urls_expand = open("Urls_Uniq_Expanded.txt", "r")
    file_carbondate = open("Urls_carbondate.txt", "w")
    current_date = datetime.datetime.now()
    for line in file_urls_expand:
        try:
            command = "http://localhost:8888/cd/"
            command = command + line.rstrip().split("|||")[0]
            print(command)
            response = requests.get(command)
            if response.status_code == 200:
                print(response.text)
                carbon_date_result = ast.literal_eval(response.text)
                creation_date = carbon_date_result["estimated-creation-date"]
                creation_date = datetime.datetime.strptime(creation_date, "%Y-%m-%dT%H
                    :%M:%S")
                print(creation_date)
                print(current_date)
                print((current_date - creation_date).days)
                file_carbondate.write(line.rstrip().split("|||")[0] + "|||" + str((
                    current_date - creation_date).days) + "\n")
        except Exception as e:
            continue
    file_carbondate.close()
    file_urls_expand.close()
```

1. **Carbon Date**: All the urls were supplied to CarbonDate to find the earliest creation date from web archives. The earliest creation date returned was substracted with current date to know the number of days since the creation of the url. I could find creation dates for 1448 urls from 2088 urls.

2. **Histogram**: 1328 urls had a carbon date between 0 to 2000 days. 97 urls had a carbon date of 2000 to 4000 days. 12 urls had a carbon date of 4000 to 6000 days. 11 urls had a carbon date og 6000- 8000 days.

Problem 3 continued on next page...

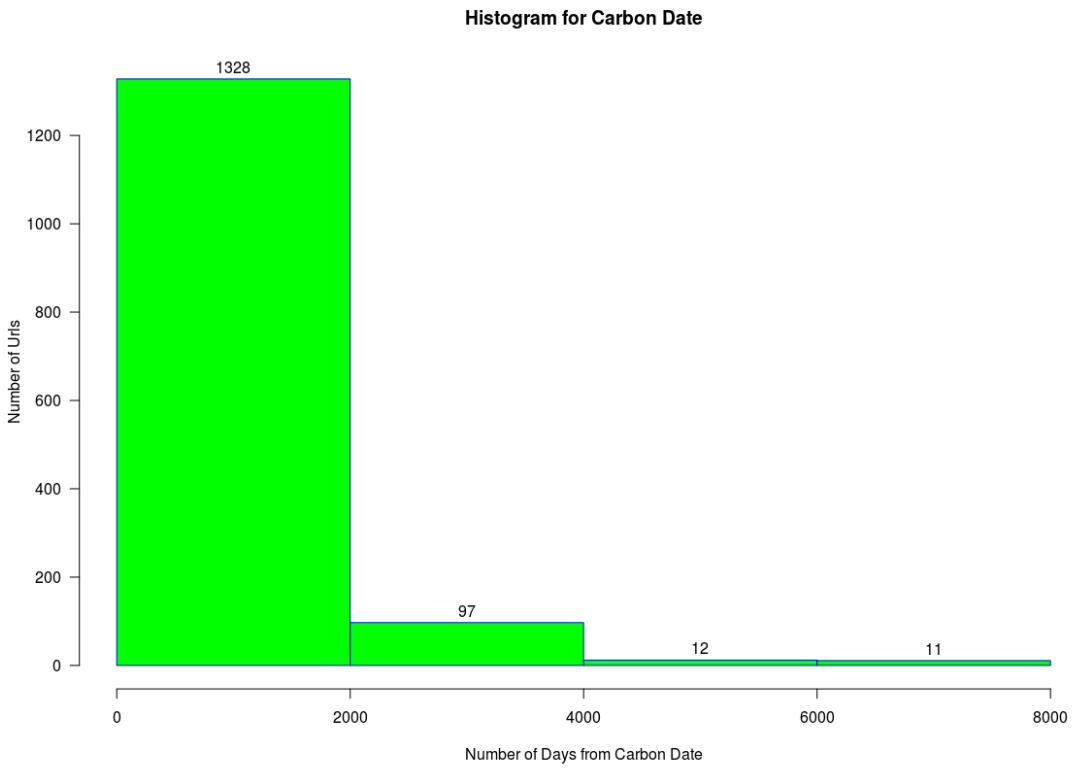

Figure 2: Histogram for Carbon Date Distribution

Total URI	2088
No Mementos	392
No date estimate	640