|  |
| --- |
|  |
| CS432 Spring 2018 |
| Assignment 2 |
|  |
| **Jonathan-Kruszewski** |
| **1/28/2018** |

|  |
| --- |
|  |

**Part 1:**

1. Write a Python program that extracts 1000 unique links from

Twitter. Omit links from the Twitter domain (twitter.com). You might want to take a look at:

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:

To accomplish this task I have utilized two python programs, “GetTwitterURLs.py” and “ModifyURLs.py”. “GetTwitterURLs.py” uses code found on <https://stackoverflow.com/questions/42401931/extract-1000-uris-from-twitter-using-tweepy-and-python> to extract urls from tweets, the code was modified to increase the number of urls extracted and to print them to a file “InitialURLList.txt”. The max number of urls to be extracted was set to 100000000 and allowed to run overnight, after 8 hours roughly 180000 urls had been extracted. From there we run “ModifiedURLs.py” which finds and removes urls to twitter and issues a get request allowing redirection to each url. Any that return a satus\_code that doesn’t equal 200 was discarded, those with status\_code equal to 200 are written to “ModifiedURLList.txt”, a counter was added to only add 1000 urls to “ModifiedURLList.txt”

“GetTwitterURLs.py” code snippet:

count = 100000000 # moved outside of class definition to avoid getting reset

class StdOutListener(StreamListener):

def on\_data(self, data):

decoded = json.loads(data)

global count # get the count

if count <= 0:

import sys

sys.exit()

else:

try:

for url in decoded["entities"]["urls"]:

print(count, ':', "%s" % url["expanded\_url"] + "\r\n")

print("%s" % url["expanded\_url"], file=open("InitialURLList.txt", "a"))

count -= 1

except KeyError:

print(decoded.keys())

def on\_error(self, status):

print(status)

if \_\_name\_\_ == '\_\_main\_\_':

l = StdOutListener()

auth = OAuthHandler(consumer\_key, consumer\_secret)

auth.set\_access\_token(access\_token, access\_token\_secret)

stream = Stream(auth, l)

stream.filter(track=['Olympics', 'Football', 'WorldCup', 'Soccer', 'Sports'])

“ModifiedURLs.py” code snippet:

tString = "https://twitter.com"

inCount = 1

with open(inputFile, "r") as ins:

urlInput = []

for line in ins:

if line[:19].lower() != tString:

urlInput.append(line)

#print("Reading input: ", inCount)

#inCount += 1

.

.

.

for url in urlList:

print("checking urls for status code and expanded url", urlCounter)

urlCounter += 1

try:

r = requests.get(url, verify=False, allow\_redirects=True)

if int(r.status\_code) == 200:

urlListExpanded.append(r.url)

goodUrlCounter += 1

print("url added to list, Status Code: ", r.status\_code, "total urls added: ", goodUrlCounter)

print("\t", r.url)

else:

badUrlCounter += 1

print("url not added to list, Status Code: ", r.status\_code, "total rejected urls: ", badUrlCounter)

except:

genericErrorInfo()

badUrlCounter += 1

print("Error Occurred, URL not Added to List total rejected urls: ", badUrlCounter)

continue

urlListExpanded = list(set(urlListExpanded))

if len(urlListExpanded) == 1000:

break

**Part 2:**

2. Download the TimeMaps for each of the target URIs. We'll use the ODU

Memento Aggregator, so for example:

URI-T = http://memgator.cs.odu.edu/timemap/json/http://www.cs.odu.edu/

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.

To accomplish this task I used “GetJSONFindMementos.py” and R, the R code can be found in “Temp.txt”. “GetJSONFindMementos.py” opens the file “ModifiedURLList.txt” and for each url and prepends the URI-T listed above to each url and retrieves the json information for each url memento. Any that returns a status\_code not equal to 200 is considered to have no mementos. The total number of each unique value of mementos is stored and output to the screen.

“GetJSONFindMementos.py” code snippet:

for url in urlInput:

mem = requests.get(urlHead+url)

if mem.status\_code != 200:

memList.append(0)

print(urlCounter, ": ", mem.status\_code)

urlCounter += 1

else:

jsonFileName = "URL #" + str(urlCounter) + ".json"

jResponse = json.loads(mem.text)

jsonList.append(jResponse)

print(jResponse, file=open(jsonFileName, "w+"))

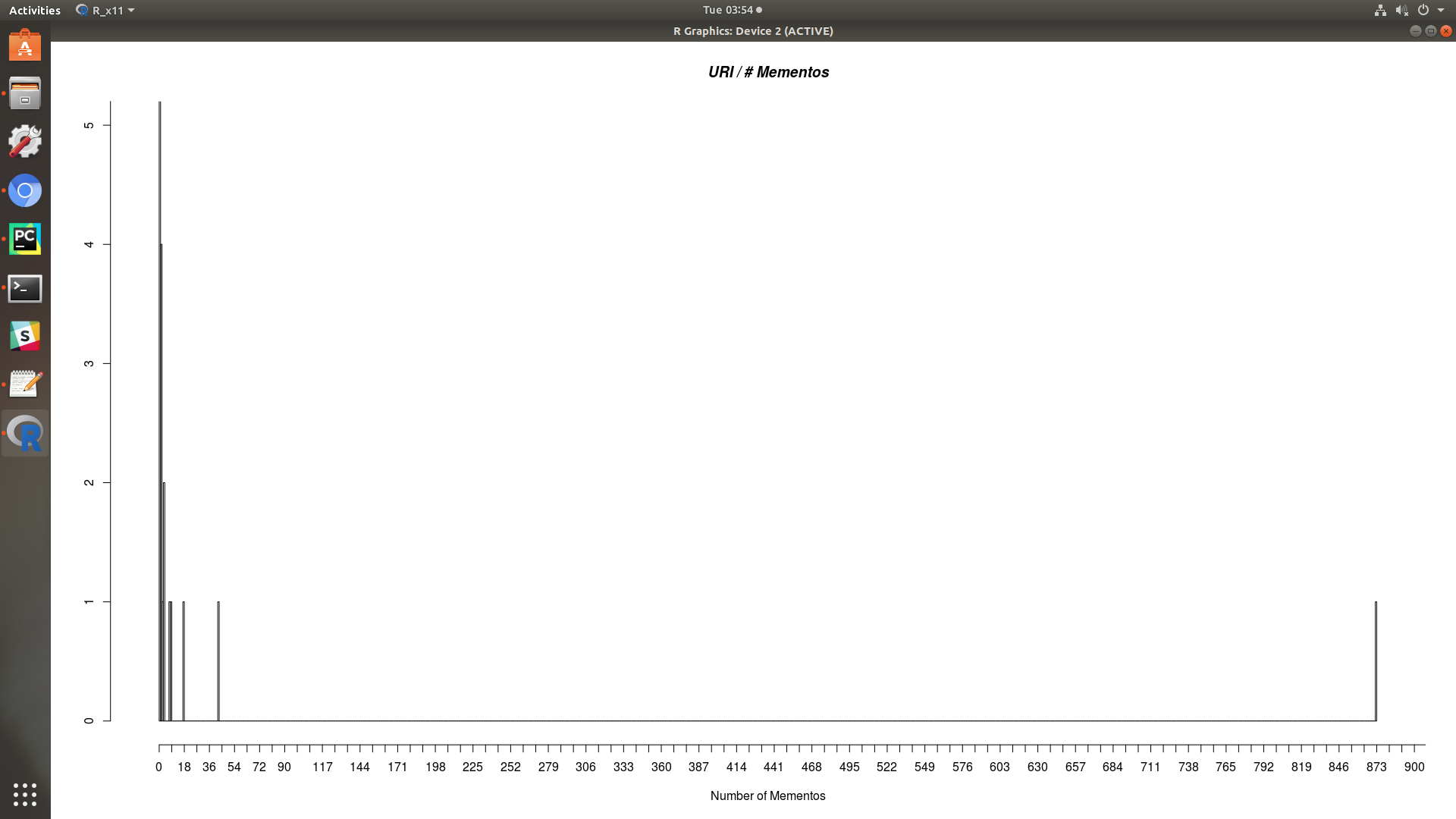
print(urlCounter, ": ", mem.status\_code, "\n", jResponse)

urlCounter += 1

for jEntry in jsonList:

memList.append(len(jEntry['mementos']['list']))

|  |
| --- |
|  |



**Part 3:**