HW6 CS432

Ondra Torkilson

November 16, 2020

Q1

Both scripts for question one and two are pictured at the end of this report. Buckle up before looking at the script for question one. My original plan was to split that script based on the first set of questions regarding requests, and the second regarding domains. Unfortunately, I could not for the life of me figure out how to get the data into a format that I could process from the file that I saved from the first part of question 1. So . . I ended up reusing the original script, and commenting out the requests portions, and adding the code to extract the domains. I am also missing the file requested in the second portion of question one which was supposed to contain the unique domain, the number of times it appeared in the data set, and the number of tweets the domain appeared in. I was able to get a list of the unique domains. This project required a lot of data manipulation that I needed more time to think about. As a feedback, I think that this project could probably use two weeks instead of one, but maybe my schedule was just too busy this week. It could also be that my brain was not in top shape this week. Sometimes that happens.

To start, in question one part one, I imported the data from the file using csv and zipped it into a python dict with the domain as the key and the number of tweets as the value. I used requests response.history to see whether the URI had redirected, and a list to track the number of redirects. I used the response.ok atrribute to determine whether the URI returned anything other than a 200 status code and lists to keep track of errors and okays. For each of the original domains, I saved the original URI, number of tweets, final URI, and status code to the outfile. Figure 1 shows the results of the redirects and responses status on the set. Eight-hundred and forty one of the 1679 were redirected. Four-hundred and ten returned in a 400 or 500 status, and 1269 returned 200.

For the second half of question one, as I mentioned earlier, I commented out the lines dealing with responses, and I added a few lines to extract the domain. I needed to check for NaNs on the tweet frequency, as some had no value, and I set that to zero. Then, if the domain was already in the dictionary for unique domains, I updated the value by adding the new tweet frequency to

the current value which represented the current tweet frequency. Figures 2 and 3 show the top five domains with the most tweets, and the number of unique domains respectively.

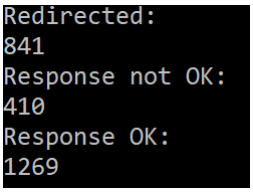

Figure 1: Redirects and Responses

Figure 2: Top Five Unique Domains with Most Tweets

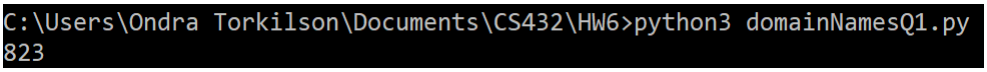

Figure 3: Number of Unique Domains

Q2

Question two was much less work than question one. I imported the three files of domain names into lists, normalized them by stripping newline characters, and made sure they were all lowercase for easier comparisons. Then, I just stepped through each list, and searched another list for the same domain. If it was a match, I added it to a new list of shared domains, and if it was not, I passed. In the end, there were 8 shared domains between the three data sets. They were either political propoganda (presstv.com) or opinon-based, biased "news" sites (breitbart.com.) Some were even conspiracy sites like activistpost.com. All of these sites share one thing in common. They do not report facts. They report opinons and bias. There may be some facts sprinkled in, but if you are looking for what we historically consider "news," you will not find much of it on these sites. Figure 4 shows the unique domains shared between the three sets of data along with the table showing the breakdown of domains shared between two sets of data at a time.

```
C:\Users\Ondra Torkilson\Documents\CS432\HW6>python3 q2.py
['beforeitsnews.com',
 '21stcenturywire.com',
 'activistpost.com',
 'dcclothesline.com',
 'globalresearch.ca',
 'intellihub.com',
 'breitbart.com',
 'presstv.com']
Data Set      Common Domains
----------  ----------------
D1D2                      41
D1D3                      12
D2D3                      22
D1D2D3                     8
```

Figure 4: Domains Shared b/t D1, D2, and D3 and Table Comparison of Data Sets

Python Scripts

```
import csv
import requests
from re import search
import pandas as pd
import itertools
from urllib.parse import urlparse
import pprint
import operator
import math

pp = pprint.PrettyPrinter(width=41, compact=True)
D2file = pd.read_csv("expanded-URLs.txt",header=None)
originalD2 = dict(zip(D2file[0].values,D2file[1].values))
#print(originalD2)

#N = 20
#testData = dict(itertools.islice(originalD2.items(), N))
#[print(key, value) for key, value in testData.items()]

#redirectedList = []
#responseErrorList = []
#repsponseOkList = []
#finalData = []

q1p2 = {}
with open('q1p2.txt', 'w') as outfile:
    for key in originalD2:
        try:
            response = requests.get(str(key), timeout=30.0, allow_redirects=True)
            finalURI = response.url
            domain = urlparse(finalURI).netloc
            toAdd = originalD2.get(key)
            if math.isnan(toAdd) == True:
                toAdd = 0
            else:
                pass
            current = q1p2.get(domain)
            if domain in q1p2:
                updatedCount = int(toAdd) + int(current)
                q1p2.update({domain:updatedCount})
            else:
                q1p2.update({domain:toAdd})
            #statusCode = response.status_code
            #print(statusCode)
            #print(finalURI)
            #insideList = [key, int(originalD2.get(key)), finalURI, str(statusCode)]
            #finalData.append(insideList)
```

```
        except requests.exceptions.RequestException as e:
            pass
    sorted_q1p2 = sorted(q1p2.items(), key=operator.itemgetter(1))
    pp.pprint(sorted_q1p2)
    outfile.write('\n'.join(str(line) for line in q1p2))

        #if (len(response.history)) != 0:
            #redirectedList.append(key)
        #if (response.ok) == False:
            #responseErrorList.append(key)
        #if (response.ok) == True:
            #repsponseOkList.append(key)
    #outfile.write('\n'.join(str(line) for line in finalData))
#[print(i) for i in finalData]
#print("Redirected: ")
#print(len(redirectedList))
#print("Response not OK: ")
#print(len(responseErrorList))
#print("Response OK: ")
#print(len(repsponseOkList))
```

```
import pprint
from tabulate import tabulate

pp = pprint.PrettyPrinter(width=30, compact=True)

with open('D2domains.txt') as f:
  raw = [line.rstrip('\n') for line in f]
#still had some domains with www, so strip that
D2domains = [line.strip('www.') for line in raw]

with open('D1domains.txt') as f:
  D1domains = [line.rstrip('\n') for line in f]

with open('D3domains.txt') as f:
  raw = [line.rstrip('\n') for line in f]
#convert to lowercase for comparisons
D3domains = [line.lower() for line in raw]

#checking lists for uniformity and correct content
#pp.pprint(D2domains)
#pp.pprint(D1domains)
#pp.pprint(D3domains)

#count common domains between D1 and D2
commonDomainsD1D2 = []
for item in D1domains:
    if item in D2domains:
        commonDomainsD1D2.append(item)
    else:
        pass
#check common domains b/t D1 and D2
#pp.pprint(commonDomainsD1D2)

#count common domains between D1 and D3
commonDomainsD1D3 = []
for item in D1domains:
    if item in D3domains:
        commonDomainsD1D3.append(item)
    else:
        pass
#check common domains b/t D1 and D2
#pp.pprint(commonDomainsD1D3)

#count common domains between D2 and D3
commonDomainsD2D3 = []
for item in D3domains:
    if item in D2domains:
        commonDomainsD2D3.append(item)
    else:
        pass
```

```
#check common domains b/t D2 and D3
#pp.pprint(commonDomainsD2D3)

#count common domains between D1, D2, and D3
commonDomainsD1D2D3 = []
for item in commonDomainsD1D2:
    if item in commonDomainsD2D3:
        commonDomainsD1D2D3.append(item)
    else:
        pass
pp.pprint(commonDomainsD1D2D3)

print(tabulate([['D1D2', len(commonDomainsD1D2)],
                ['D1D3', len(commonDomainsD1D3)],
                ['D2D3', len(commonDomainsD2D3)],
                ['D1D2D3', len(commonDomainsD1D2D3)]],
               headers=['Data Set', 'Common Domains']))
```