HW6 - Analyzing Disinformation Domains

Brenden Lewis
Due: 11/15/2020 11:59PM

Q1

Listing 1 below shows the code used to read in and process the D2 dataset. I decided it would be easiest to collect and process all URIs first before extracting and identifying unique domains. I used the *requests* library to get the HTTP status code and final uri for each URI in the dataset; this processing was handled in the *processHTTPStatus()* method that takes a URI as an argument.

```
1  import tweepy
2  from tweepy.streaming import StreamListener
3  from tweepy import OAuthHandler
4  from tweepy import Stream
5  import requests
6  from requests.exceptions import ConnectionError
7  import csv
8  import json
9  from operator import itemgetter
10 from urllib.parse import urlparse
11 import socket
12 import ast
13
14 allURIs = []    #  {'OG_URI': og, 'FREQUENCY': count, 'FINAL_URI':
      final, 'STATUS': http}
15 uniqueDomains = []  #  {'DOMAIN': domain, 'NUM_IN_DATA':dataCount,'
      NUM_IN_TWEETS':tweetCount}
16 sortedDomains = []
17 diff_URI_Count = 0
18 status200 = 0
19 status404 = 0
20 count = 1
21
22
23
24 def process():
25     #loop_count = 0
26     with open("expanded-URLs.csv",'r') as f:
27         for line in csv.DictReader(f):
28             #if(loop_count==100):
29                 #break
30             uri = line['Article URL']
31             freq = line['total_freq']
```

```
32              final_uri, status = processHTTPStatus(uri)
33              all_dict = {'OG_URI': uri, 'FREQUENCY': freq, 'FINAL_URI':
    final_uri, 'STATUS': status}
34              allURIs.append(all_dict)
35              #loop_count+=1
36
37      for item in allURIs:
38          item_uri = item['FINAL_URI']
39          item_freq = item['FREQUENCY']
40          extractDomain(item_uri,item_freq)
41
42      global sortedDomains
43      sortedDomains = sorted(uniqueDomains,key=itemgetter('NUM_IN_DATA'),
        reverse=True)
44
45
46  def extractDomain(uri,frequency):
47      domain = urlparse(uri).netloc
48      domain_dict = {'DOMAIN':domain,'NUM_IN_DATA':1,'NUM_IN_TWEETS':
    frequency}
49      exists = False
50      exists_index = 0
51
52      for index, dom in enumerate(uniqueDomains):
53          if domain == dom['DOMAIN']:
54              exists = True
55              exists_index = index
56
57
58      if exists == False:
59          uniqueDomains.append(domain_dict)
60      else:
61          uniqueDomains[exists_index]['NUM_IN_DATA']+=1
62
63
64  def processHTTPStatus(uri):
65      global status200, status404, diff_URI_Count, count
66      final_url=""
67      status=""
68      print("Processing URI "+str(count)+": "+uri)
69      try:
70          response = requests.get(uri,timeout=20)
71          final_url = response.url
72          status = response.status_code
73          if(final_url != uri):
74              diff_URI_Count+=1
```

```
75      except (ConnectionError, requests.exceptions.Timeout, requests.
    exceptions.ReadTimeout, requests.exceptions.TooManyRedirects):
76          print("Connection Error")
77          final_url = uri
78          status = 0
79      except socket.timeout:
80          print("Timeout Error")
81          final_url = uri
82          status = 0
83
84      if status == 200:
85          status200+=1
86      else:
87          status404+=1
88
89      print("Done")
90      count+=1
91
92      return(final_url, status)
93
94  def write_URIs_To_File():
95      with open("ProcessedURIs.txt","w") as f:
96          for item in allURIs:
97              f.write(str(item)+"\n")
98
99  def write_Domain_To_File():
100     with open("UniqueDomainsForProc.txt","w") as f:
101         #f.write("Number of Unique Domains: "+str(len(sortedDomains))
    +"/"+str(len(allURIs))+"\n\n")
102         for item in sortedDomains:
103             f.write(str(item)+"\n")
104             #f.write("Domain: "+item['DOMAIN']+"\n")
105             #f.write("Frequency in Data: "+str(item['NUM_IN_DATA'])+"\n
    ")
106             #f.write("Frequency in Twitter Posts: "+str(item['
    NUM_IN_TWEETS'])+"\n\n")
107
108
109
110 def outputURIs():
111     global status200, status404, diff_URI_Count
112     print("\n")
113     for item in allURIs:
114         print("URI:"+str(item["OG_URI"]))
115         print("Frequency:"+str(item["FREQUENCY"]))
116         print("Final URI:"+str(item["FINAL_URI"]))
117         print("Status:"+str(item["STATUS"]))
```

```
118         print("\n")
119
120     print("\n")
121
122 def outputDomains():
123     print("Unique Domains: \n")
124     for item in uniqueDomains:
125         print("Domain: "+item['DOMAIN'])
126         print("Data Frequency: "+str(item['NUM_IN_DATA']))
127         print("Tweet Frequency: "+str(item['NUM_IN_TWEETS']))
128         print("\n")
129     print("\n")
130
131
132 def outputStats():
133     global status200, status404
134     print("URIs with 200 response: "+str(status200))
135     print("URIs with 404 response: "+str(status404))
136     print("URIs redirected to different URIs: "+str(diff_URI_Count)+"\n
    ")
137
138     #print("Unique Domains: "+str(unique_domain_count))
139     print("Unique Domains: "+str(len(uniqueDomains)))
140     #print("Unique Domain List Size: "+ str(len(uniqueDomains)))
141
142 def readURIsForDomains():
143     with open("ProcessedURIs.txt",'r') as f:
144         for line in f:
145             new_dict = ast.literal_eval(line)
146             uri = new_dict['FINAL_URI']
147             freq = new_dict['FREQUENCY']
148             extractDomain(uri,freq)
149     global sortedDomains
150     sortedDomains = sorted(uniqueDomains,key=itemgetter('NUM_IN_DATA'),
    reverse=True)
151
152 if __name__=='__main__':
153     process()
154     #readURIsForDomains()
155     #outputURIs()
156     #outputDomains()
157     write_URIs_To_File()
158     write_Domain_To_File()
159     outputStats()
```

Listing 1: Python code for Q1

Once a URi was finished processing, a dictionary was created storing the original URI, its fre-

quency in Twitter posts, final URI, and status code returned from *requests.get*. An example of this structure is shown in Listing 2 below. Each dictionary per URI was stored in ProcessingURIs.txt; this allowed me to more easily test other code without having to re-process the URIs as it is a lengthy process at about 2 hours.

```
{'OG_URI': 'http://newsteller.org', 'FREQUENCY': '2', 'FINAL_URI': '
   http://newsteller.org/', 'STATUS': 200}
```

Listing 2: Entry from ProcessedURIs.txt

I utilized *urlparse(uri).netloc* to extract the domain of a given uri. With the extracted domain and the passed in tweet frequency, I was created and stored dictionaries containing the domain, the number of times it appeared within the dataset, and tweet frequency in a dictionary list. Before they could be stored, however, I had to check first if the domain was already present in the list; if the domain was already present, the *NUM_IN_DATA* value would need to be iterated by one. This was done simply by enumerating the list and using an index to access the domain and its values directly.

Listing 3 below shows the statistics for URI processing. Out of a total of 1679 URIs, 1186 resulted in a 200 status code while 493 resulted in a status code other than 200 (almost entirely 404, but there were one or two alternate codes). 793 of the URIs were redirected to a different URI.

```
URIs with 200 response: 1186
URIs with 404 response: 493
URIs redirected to different URIs: 793

Unique Domains: 812
```

Listing 3: Stats from URI processing

Listing 4 shows a snippet of the unique domains sorted by their frequency in the data set. There were a total of 812 unique domains out of the 1679 URIs processed. Each listing shows the domain, its frequency in the data set, and the number of tweets they were featured in. Interestingly enough, among the most common domains in the data set were Google domains, with *news.google.com* being the most prominent.

However, Google did not appear in the most amount of tweets in the set. Listing 5 shows the top 5 domains with the largest presence in twitter posts. Of the top listed domains, there are conspiracy theory sites, a US news outlet, a British newspaper, and a Middle Eastern news outlet. The domain that appeared in the most tweets was *unz.com* with 514 tweets. The domain refers to an alternative media news outlet, discovered to be pushing conspiracy theories, anti-semitism, and whitee supremacy.

```
Number of Unique Domains: 812/1679 URIs

Domain: news.google.com
Frequency in Data: 40
Frequency in Twitter Posts: 1
```

```
6
7 Domain: www.newslocker.com
8 Frequency in Data: 38
9 Frequency in Twitter Posts: 1
10
11 Domain: 21stcenturywire.com
12 Frequency in Data: 36
13 Frequency in Twitter Posts: 29
14
15 Domain: www.google.com
16 Frequency in Data: 28
17 Frequency in Twitter Posts: 1
18
19 Domain: news.quiboat.com
20 Frequency in Data: 22
21 Frequency in Twitter Posts: 1
22
23 Domain: clarityofsignal.com
24 Frequency in Data: 20
25 Frequency in Twitter Posts: 464
```

Listing 4: Segment of unique domains sorted by frequency in data set

From the data set, the top 5 domains that appeared in the most tweets were:

```
1. unz.com [514]
2. clarityofsignal.com [464]
3. nbcnews.com [316]
4. middleeasteye.net [279]
5. theguardian.com [242]
```

Listing 5: Top 5 domains with most tweets

Q2

Listing 6 shows the code used for Q2. To keep everything simple, I read in each data set and stored each item as a dictionary in seperate lists. To compare each data set, I used a set of nested for-loops (one for each data set) to compare the stored domains; because I forgot to do so in Q1, I had to use a regular expression to remove the ”www.” present in some domains in D2, and lower case the domains in D3 for even comparison. If any domains were equal across the data sets, they were add to a list of domains for the respective data set pairing. To ensure no duplicates, an additional check to the list was used to check if the domain was already present in the list.

```
1 import matplotlib
2 import matplotlib.pyplot as plt
3 import pandas as pd
```

```
4  from operator import itemgetter
5  import ast
6  import csv
7  import re
8
9  d1 = []
10 d2 = [] #From Q1
11 d3 = []
12
13 domainsInD1_D2 = []
14 domainsInD2_D3 = []
15 domainsInD1_D3 = []
16
17 domainsInAll = []
18
19
20 def readD1():
21     with open("D1.csv",'r') as f:
22         for line in csv.DictReader(f):
23             d1_dict = {'DOMAIN':line["Domain"]}
24             d1.append(d1_dict)
25
26 def readD2():
27     with open("UniqueDomainsForProc.txt",'r') as f:
28         for line in f:
29             d2_dict = ast.literal_eval(line)
30             d2.append(d2_dict)
31
32 def readD3():
33     with open("D3.csv",'r') as f:
34         for line in csv.DictReader(f):
35             d3_dict = {'DOMAIN':line["Domain"],'COUNTRY':line['Country'
    ]}
36             d3.append(d3_dict)
37
38 def compareDomains():
39     for d1Item in d1:
40         dom1 = d1Item['DOMAIN']
41         #print(dom1+'\n')
42         for d2Item in d2:
43             dom2 = d2Item['DOMAIN']
44             if dom2.startswith('www.'):
45                 dom2 = re.sub(r'www.','',dom2)
46             #print(dom2+'\n')
47             if (dom1 == dom2):
48                 if dom2 not in domainsInD1_D2:
49                     domainsInD1_D2.append(dom2)
```

```
50
51              for d3Item in d3:
52                  dom3 = d3Item['DOMAIN']
53                  dom3 = dom3.lower()
54
55                  if (dom1 == dom3):
56                      if dom3 not in domainsInD1_D3:
57                          domainsInD1_D3.append(dom3)
58                  if (dom2 == dom3):
59                      if dom3 not in domainsInD2_D3:
60                          domainsInD2_D3.append(dom3)
61                  if (dom1 == dom2 and dom2 == dom3):
62                      if dom3 not in domainsInAll:
63                          domainsInAll.append(dom3)
64
65
66  def drawTable():
67      sorted12 = sorted(domainsInD1_D2)
68      sorted23 = sorted(domainsInD2_D3)
69      sorted13 = sorted(domainsInD1_D3)
70      sortedAll = sorted(domainsInAll)
71
72
73      table1 = pd.DataFrame({
74          'D1 & 2': sorted12,
75      })
76
77      table2 = pd.DataFrame({
78          'D2 & 3': sorted23,
79      })
80
81      table3 = pd.DataFrame({
82          'D1 & 3': sorted13,
83      })
84
85      table4 = pd.DataFrame({
86         'In All': sortedAll,
87      })
88
89      print(table1)
90      print("\n")
91      print(table2)
92      print("\n")
93      print(table3)
94      print("\n")
95      print(table4)
96
```

```
if __name__=='__main__':
    readD1()
    readD2()
    readD3()
    compareDomains()
    drawTable()
```

Listing 6: Python code for Q2

Listings 7-9 below show tables of each set-comparison with all the common domains. Listing 10 shows a comparison table between all data sets with the domains present all three. Interesting to note, the top domains across all domains nearly matches the top domains between every other comparison. Additionally, nearly all the common domains are conspiracy theory sites.

```
                          D1 & 2
0              21stcenturywire.com
1               abovetopsecret.com
2                 activistpost.com
3               beforeitsnews.com
4              blacklistednews.com
5                    breitbart.com
6                      cbsnews.com
7                          cnn.com
8                 dailymail.co.uk
9               dcclothesline.com
10       fellowshipoftheminds.com
11                     foxnews.com
12  fuhrerious88blog.wordpress.com
13              globalresearch.ca
14                       heavy.com
15                    infowars.com
16                  intellihub.com
17         investmentwatchblog.com
18     landdestroyer.blogspot.com
19                lewrockwell.com
20                   mirror.co.uk
21                    nbcnews.com
22                   news.sky.com
23                nydailynews.com
24                    nytimes.com
25                    presstv.com
26                         rt.com
27                sputniknews.com
28               theantimedia.org
29              thedailybeast.com
30            thedailysheeple.com
31          theeventchronicle.com
32       thefreethoughtproject.com
```

```
35 33                theguardian.com
36 34               theintercept.com
37 35        themillenniumreport.com
38 36             therussophile.org
39 37                   thestar.com
40 38          thetruthseeker.co.uk
41 39                       upi.com
42 40            veteranstoday.com
43 41           washingtonpost.com
44 42                worldtruth.tv
45 43             yournewswire.com
```

Listing 7: Domains present in both D1 and D2

```
1                       D2 & 3
2 0         21stcenturywire.com
3 1            activistpost.com
4 2           beforeitsnews.com
5 3              breitbart.com
6 4    collective-evolution.com
7 5              davidicke.com
8 6           dcclothesline.com
9 7          de.sputniknews.com
10 8             deutsch.rt.com
11 9          fr.sputniknews.com
12 10           gellerreport.com
13 11          globalresearch.ca
14 12          humansarefree.com
15 13               infowars.com
16 14             intellihub.com
17 15           off-guardian.org
18 16                presstv.com
19 17       ronpaulinstitute.org
20 18               rubikon.news
21 19                   sott.net
22 20               theduran.com
23 21  thewashingtonstandard.com
24 22               ukcolumn.org
25 23              worldtruth.tv
```

Listing 8: Domains present in both D2 and D3

```
1                  D1 & 3
2 0   21stcenturywire.com
3 1      activistpost.com
4 2     beforeitsnews.com
5 3         breitbart.com
6 4     dcclothesline.com
7 5     globalresearch.ca
```

```
8  6          infowars.com
9  7        intellihub.com
10 8           presstv.com
11 9       wakingtimes.com
12 10        worldtruth.tv
13 11        zerohedge.com
```

Listing 9: Domains present in both D1 and D3

```
1                 In All
2  0  21stcenturywire.com
3  1     activistpost.com
4  2    beforeitsnews.com
5  3        breitbart.com
6  4    dcclothesline.com
7  5    globalresearch.ca
8  6         infowars.com
9  7       intellihub.com
10 8          presstv.com
11 9        worldtruth.tv
```

Listing 10: Domains present in all data sets

Q3

I opted to use two different Python scripts to handle Q3: one to gather the tweets featuring domain links common between the D2 and D3 data set (Listing 11), and one to analyze the gathered tweets and create the appropriate graphs (Listing 13).

For gathering tweets, I used Tweepy. For the sake of time, I limited the number of tweets per domain to a maximum of 200 tweets. At the same time, I filtered out any retweets using *"-filter:retweets"* in the Cursor call. Each tweet was stored as a dictionary with its ID, the account that sent the tweet, the time it was tweeted, the domain featured in the tweet, the link to said domain as it appears in the text, and the full text in the tweet.

```
1  import tweepy
2  from tweepy.streaming import StreamListener
3  from tweepy import OAuthHandler
4  from tweepy import Stream
5  import json
6  import time
7  import traceback
8  import math
9
10 domains = []
11 tweets = []
```

```
domainStats = []    # {DOMAIN:domain, NUM_TWEETS:numTweets, NUM_ACCOUNT:numAccounts}
accounts = []

LIMIT = 200
totalTweetCount = 0
lowestTime = 0
highestTime = 0


# Keys ommitted
consumer_key="***"
consumer_secret="***"
access_token="***"
access_secret="***"

# Handles authorization with Twitter
auth = OAuthHandler(consumer_key,consumer_secret)
auth.set_access_token(access_token,access_secret)
api = tweepy.API(auth, wait_on_rate_limit=True, wait_on_rate_limit_notify=True)
#api = tweepy.API(auth)

def readInDomains():
    with open("D1_D3_Table.txt",'r') as f:
        next(f)
        for line in f:
            (key,value) = line.split()
            domains.append(value)


def getTweets():
    global totalTweetCount

    for domain in domains:
        print("Fetching: "+domain)
        activeTweetCount = 0    #Number of tweets per domain
        for tweet in tweepy.Cursor(api.search, q="url:"+ domain +" -filter:retweets", lang="en").items(LIMIT):
            print("Tweet "+str(activeTweetCount+1))

            for url in tweet.entities["urls"]:
                tweet_ID = tweet.id_str
                tweet_Account = tweet.user.screen_name
                tweet_Time = tweet.created_at.strftime("%Y%m%d%H%M%S")
                tweet_Link = tweet.entities["urls"][0]["expanded_url"]
                tweet_Text = tweet.text
```

```
56
57                  tweet_dict = {'ID':tweet_ID,'ACCOUNT':tweet_Account,'
    TIME':tweet_Time,'DOMAIN':domain,
58                  'LINK':tweet_Link,'TEXT_BODY':tweet_Text}
59
60                  tweets.append(tweet_dict)
61                  totalTweetCount+=1
62
63              activeTweetCount+=1
64
65          print("Domain Closed"+"\n")
66
67 def exportToJSON():
68     with open("tweets.json",'w') as f:
69         json.dump(tweets, f, indent=2)
70
71 def writeStats():
72     pass
73
74 if __name__=='__main__':
75     readInDomains()
76     getTweets()
77     exportToJSON()
```

Listing 11: Python code for gathering tweets for Q3

Each captured tweet was exported to a JSON file called *tweets.json*. The JSON format of each tweet in the file is shown below in Listing 12. This file is processed in *q3.py*.

```
1   {
2     "ID": "1333480504297988104",
3     "ACCOUNT": "Marie61172377",
4     "TIME": "20201130183821",
5     "DOMAIN": "21stcenturywire.com",
6     "LINK": "https://21stcenturywire.com/2020/11/23/covid-19-mounting-
    evidence-of-international-fraud/",
7     "TEXT_BODY": "COVID 19: Mounting Evidence of International Fraud -
    21st Century Wire https://t.co/t6SPl6F6C5"
8   }
```

Listing 12: Example tweet gathered in JSON

The code for processing the *tweets.json*, gathering tweet and domain statistics, and drawing graphs is featured below in Listing 13. Loading the data from *tweets.json*, the data was passed into seperate functions to gather the different stats for Q3.

```
1 import json
2 import matplotlib
3 import matplotlib.pyplot as plt
```

```
4  from matplotlib.pyplot import figure
5  import seaborn as sns
6  import pandas as pd
7  from pandas import DataFrame
8  from operator import itemgetter
9
10 domains = []
11 domainStats = []    # {DOMAIN:domain, NUM_TWEETS:numTweets, NUM_ACCOUNT
       :numAccounts}
12 accounts = []
13 timeRange = []  # [min,max]
14 totalTweets = 0
15
16 def process():
17     global totalTweets
18     with open("tweets.json",'r') as f:
19         data = json.load(f)
20
21     totalTweets = len(data)
22
23     findTimeRange(data)
24     getAllAccounts(data)
25     buildDomainStats(data)
26
27
28
29 def drawBarChart():
30     #fig = plt.figure()
31     domains = []
32     numOfTweets = []
33     dataframes = []
34     for domain in domainStats:
35         dom = domain["DOMAIN"]
36         tweetCount = int(domain["NUM_TWEETS"])
37
38         domains.append(dom)
39         numOfTweets.append(tweetCount)
40         #domains.append({dom,tweetCount})
41
42     d_dict = {'Domain':domains,'Tweet Count':numOfTweets}
43
44     df = pd.DataFrame(d_dict)
45
46
47
48     figure(num=None, figsize=(20,25), dpi=80, facecolor='w', edgecolor=
       'r')
```

```
49      sns.barplot(x="Tweet Count", y="Domain", data=df)
50
51      plt.title("Number of Tweets per Domain")
52      #plt.bar(domains,numOfTweets)
53      #ax = fig.add_axes([0,0,1,1])
54      #ax.bar(domains,numOfTweets)
55      plt.show()
56
57  def drawBarChartAccounts():
58
59      domains = []
60      numOfAccounts = []
61      dataframes = []
62      for domain in domainStats:
63          dom = domain["DOMAIN"]
64          accountCount = int(domain["NUM_ACCOUNTS"])
65
66          domains.append(dom)
67          numOfAccounts.append(accountCount)
68
69
70      d_dict = {'Domain':domains,'Account Count':numOfAccounts}
71
72      df = pd.DataFrame(d_dict)
73
74
75
76      figure(num=None, figsize=(20,25), dpi=80, facecolor='w', edgecolor=
    'r')
77      sns.barplot(x="Account Count", y="Domain", data=df)
78
79      plt.title("Number of Accounts per Domain")
80      plt.show()
81
82
83  def readInDomains():
84      with open("D2_D3_Table.txt",'r') as f:
85          next(f)
86          for line in f:
87              (key,value) = line.split()
88              domains.append(value)
89
90  def findTimeRange(data):
91      times = []
92      for item in data:
93          times.append(item["TIME"])
94
```

```
95      timeRange.append(min(times))
96      timeRange.append(max(times))
97
98  def getAllAccounts(data):
99      for i in data:
100         if(i["ACCOUNT"] not in accounts):
101             accounts.append(i["ACCOUNT"])
102
103 def buildDomainStats(data):
104     for domain in domains:
105         numTweetsPerDomain = 0
106         accountsPerDomain = []
107         for item in data:
108             if(item["DOMAIN"] == domain):
109                 numTweetsPerDomain+=1
110                 if(item["ACCOUNT"] not in accountsPerDomain):
111                     accountsPerDomain.append(item['ACCOUNT'])
112
113         domain_dict = {"DOMAIN":domain, "NUM_TWEETS":numTweetsPerDomain
        , "NUM_ACCOUNTS":len(accountsPerDomain)}
114         domainStats.append(domain_dict)
115
116 def writeStats():
117     for item in domainStats:
118         print("Domain: "+item["DOMAIN"])
119         print("Number of Tweets: "+str(item["NUM_TWEETS"]))
120         print("Number of Accounts: "+str(item["NUM_ACCOUNTS"]))
121         print("\n")
122     print("Total Number of Tweets: "+str(totalTweets))
123     print("Total Number of Accounts: "+str(len(accounts)))
124     print("Time Range: {"+timeRange[0]+" --- "+timeRange[1]+"}")
125
126 def exportDomainStats():
127     sortedDomainStats = sorted(domainStats,key=itemgetter('NUM_TWEETS')
        , reverse=True)
128     with open("DomainStatsForProcessing.txt",'w') as f:
129         for item in sortedDomainStats:
130             f.write(str(item)+'\n')
131
132
133
134 if __name__=='__main__':
135     readInDomains()
136     process()
137     #drawBarChart()
138     drawBarChartAccounts()
139     #writeStats()
```

```
140     #exportDomainStats()
```

Listing 13: Python code for analyzing gathered tweets and drawing the graph for Q3

buildDomainStats() was used to gather the number of tweets and accounts for each domain. A separate, temporary list of accounts was used for comparisons to ensure multiple tweets from a single account were not being accounted for when finding total account numbers.

findTimeRange() was used to find the window of time of which all tweets gathered were posted. A list was used to contain the time for each tweet, and the *min()* and *max()* functions were used with the list as a parameter to get both the oldest and most recent times across the data. These two values were stored in a separate array for use in output.

The final stats are shown below in Listing 14. Each domain is listed with the number of tweets featuring that domain and the number of different accounts posting those tweets. Across all domains, there was a grand total of 2908 tweets gathered, with a collective number of 1156 different accounts that posted those tweets. The range of time across those tweets was November 21, 2020 12:25PM to November 30, 2020 7:29PM.

```
1  Domain: 21stcenturywire.com
2  Number of Tweets: 207
3  Number of Accounts: 88
4
5
6  Domain: activistpost.com
7  Number of Tweets: 204
8  Number of Accounts: 79
9
10
11 Domain: beforeitsnews.com
12 Number of Tweets: 69
13 Number of Accounts: 33
14
15
16 Domain: breitbart.com
17 Number of Tweets: 210
18 Number of Accounts: 122
19
20
21 Domain: collective-evolution.com
22 Number of Tweets: 205
23 Number of Accounts: 123
24
25
26 Domain: davidicke.com
27 Number of Tweets: 71
28 Number of Accounts: 52
29
```

```
30
31 Domain: dcclothesline.com
32 Number of Tweets: 202
33 Number of Accounts: 105
34
35
36 Domain: de.sputniknews.com
37 Number of Tweets: 5
38 Number of Accounts: 4
39
40
41 Domain: deutsch.rt.com
42 Number of Tweets: 14
43 Number of Accounts: 10
44
45
46 Domain: fr.sputniknews.com
47 Number of Tweets: 12
48 Number of Accounts: 11
49
50
51 Domain: gellerreport.com
52 Number of Tweets: 204
53 Number of Accounts: 81
54
55
56 Domain: globalresearch.ca
57 Number of Tweets: 0
58 Number of Accounts: 0
59
60
61 Domain: humansarefree.com
62 Number of Tweets: 207
63 Number of Accounts: 88
64
65
66 Domain: infowars.com
67 Number of Tweets: 231
68 Number of Accounts: 110
69
70
71 Domain: intellihub.com
72 Number of Tweets: 76
73 Number of Accounts: 28
74
75
76 Domain: off-guardian.org
```

```
Number of Tweets: 0
Number of Accounts: 0


Domain: presstv.com
Number of Tweets: 211
Number of Accounts: 64


Domain: ronpaulinstitute.org
Number of Tweets: 158
Number of Accounts: 81


Domain: rubikon.news
Number of Tweets: 10
Number of Accounts: 7


Domain: sott.net
Number of Tweets: 0
Number of Accounts: 0


Domain: theduran.com
Number of Tweets: 0
Number of Accounts: 0


Domain: thewashingtonstandard.com
Number of Tweets: 200
Number of Accounts: 30


Domain: ukcolumn.org
Number of Tweets: 207
Number of Accounts: 83


Domain: worldtruth.tv
Number of Tweets: 205
Number of Accounts: 28


Total Number of Tweets: 2908
Total Number of Accounts: 1156
Time Range: {20201121122509 --- 20201130192909}
```

```
124                 #11/21/2020 12:25PM --- 11/30/2020 7:29PM
```

Listing 14: Domain tweet and account statistics

Figure 1 below shows a horizonantal bar chart visualizing the tweet counts for each domain, with the domains on the y-axis and tweet counts on the x-axis. Figure 2 shows the same domains with their account counts; these were drawn using the *seaborn* visualization library. Unfortunantly, I was forced to rotate and scale them down to cleanly fit them in this report.

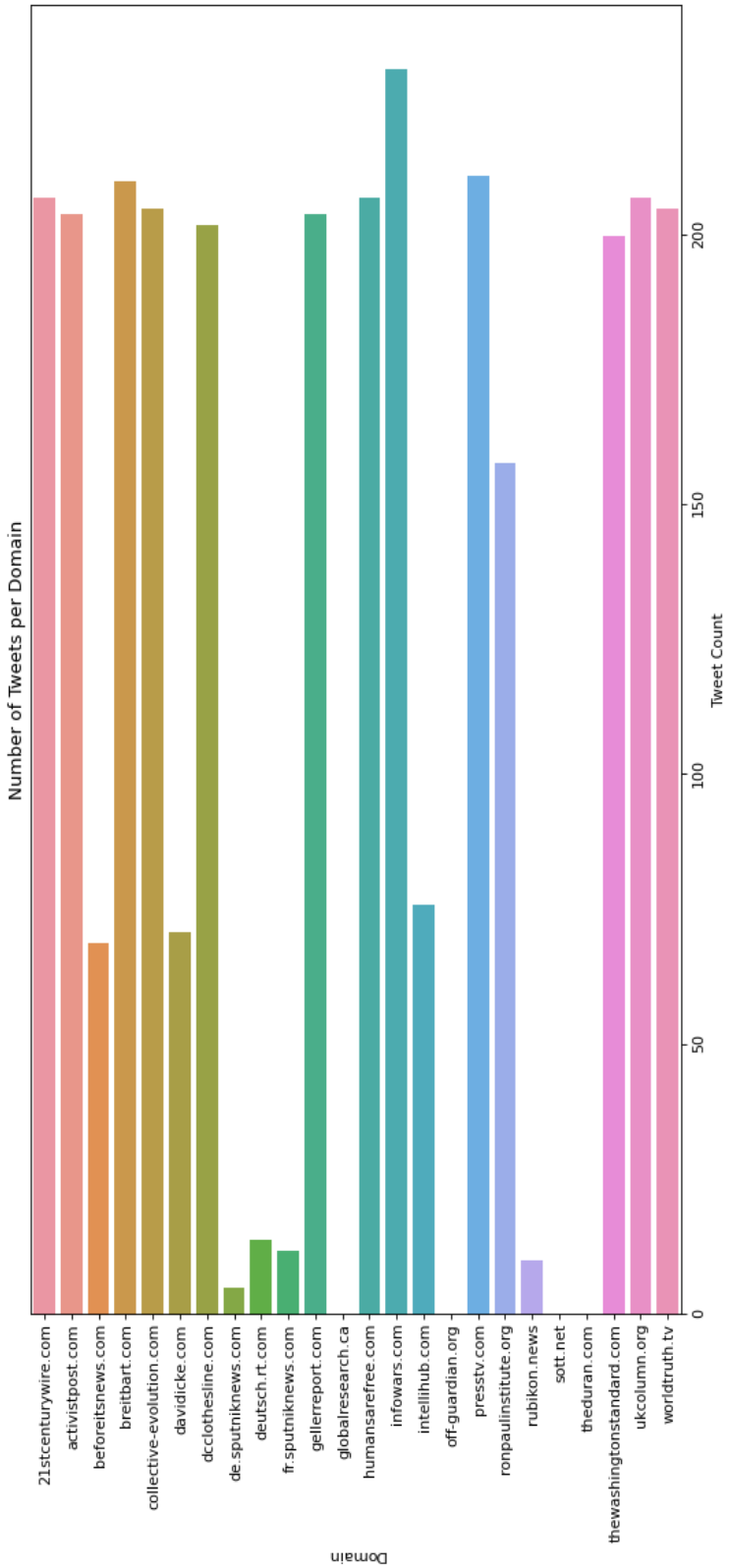

Figure 1: Graph showing the number of tweets per domain

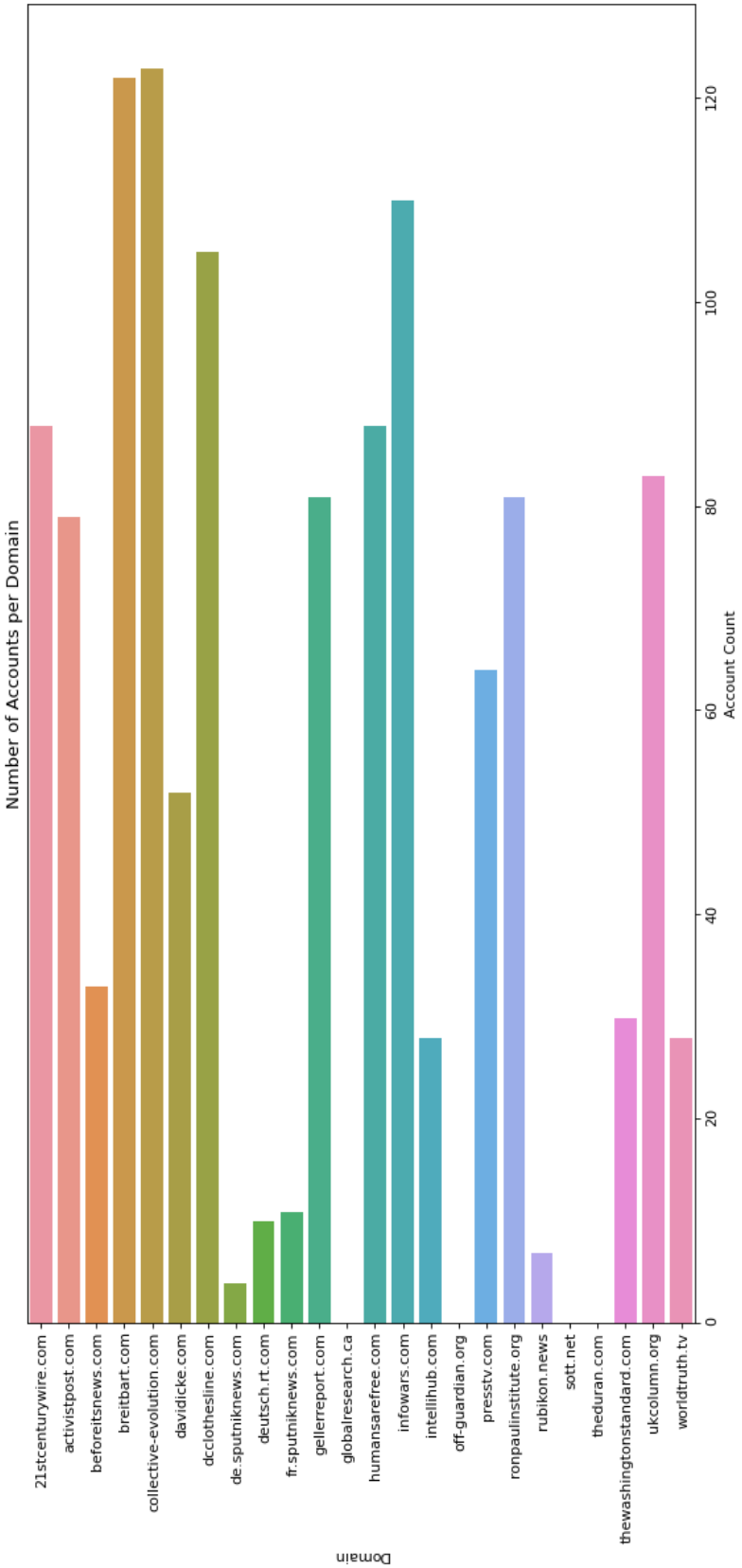

Figure 2: Graph showing the number of accounts per domain

Q4

Listing 15 below features the code used for Q4. It is very similar in structure to the code in Q2, with the biggest differences being the replacement of the D3 data set with the data collected in Q3 (labeled Q3) and the comparison logic. Since there was no need to observe overlap in all three data sets, I used two seperate, nested for-loops for the comparisons between Q3 and D1/D2.

```
1  import json
2  import matplotlib
3  import matplotlib.pyplot as plt
4  from matplotlib.pyplot import figure
5  import seaborn as sns
6  import pandas as pd
7  from pandas import DataFrame
8  import ast
9  import csv
10 from operator import itemgetter
11 import re
12
13 d1 = [] # DOMAIN, NUM_CITATIONS
14 d2 = [] # DOMAIN, NUM_IN_TWEETS
15 q3 = [] # DOMAIN, NUM_TWEETS
16
17 domainsInD1_Q3 = [] # DOMAIN, TOTAL_FREQ, D1_FREQ, Q1_FREQ
18 domainsInD2_Q3 = [] # DOMAIN, D2_FREQ, Q1_FREQ
19
20
21 def readD2():
22     global d2
23     temp = []
24     with open("UniqueDomainsForProc.txt",'r') as f:
25         for line in f:
26             d2_dict = ast.literal_eval(line)
27             temp.append(d2_dict)
28     for item in temp:
29         if item["NUM_IN_TWEETS"] == "":
30             item["NUM_IN_TWEETS"] = 0
31         else:
32             item["NUM_IN_TWEETS"] = int(item["NUM_IN_TWEETS"])
33
34     d2 = sorted(temp,key=itemgetter("NUM_IN_TWEETS"), reverse=True)
35
36 def readD1():
37     global d1
38     temp = []
39     with open("D1.csv",'r') as f:
40         for line in csv.DictReader(f):
```

```
41              if line["# Citations in our Alternative Narrative Tweets"]
    == "":
42                  d1_dict = {'DOMAIN':line["Domain"], 'NUM_CITATIONS':0}
43              else:
44                  d1_dict = {'DOMAIN':line["Domain"], 'NUM_CITATIONS':int
    (line["# Citations in our Alternative Narrative Tweets"])}
45              temp.append(d1_dict)
46
47      d1 = sorted(temp,key=itemgetter("NUM_CITATIONS"), reverse=True)
48
49  def readQ3():
50      with open("DomainStatsForProc.txt",'r') as f:
51          for line in f:
52              q3_dict = ast.literal_eval(line)
53              q3.append(q3_dict)
54
55  def compareDomains():
56      global domainsInD1_Q3, domainsInD2_Q3
57      temp1 = []
58      for d1Item in d1:
59          dom1 = d1Item['DOMAIN']
60
61          for q3Item in q3:
62              dom3 = q3Item['DOMAIN']
63
64              if (dom1 == dom3):
65                  if dom3 not in domainsInD1_Q3:
66                      d1_freq = d1Item["NUM_CITATIONS"]
67                      q3_freq = q3Item["NUM_TWEETS"]
68                      tot_freq = d1_freq + q3_freq
69                      new_dict = {"DOMAIN":dom3,"TOTAL_FREQ":tot_freq,"
    D1_FREQ":d1_freq,"Q3_FREQ":q3_freq}
70                      temp1.append(new_dict)
71
72      domainsInD1_Q3 = sorted(temp1, key = itemgetter("TOTAL_FREQ"),
    reverse=True)
73
74      temp2 = []
75      for d2Item in d2:
76          dom2 = d2Item['DOMAIN']
77          if dom2.startswith('www.'):
78              dom2 = re.sub(r'www.','',dom2)
79
80          for q3Item in q3:
81              dom3 = q3Item['DOMAIN']
82
83              if (dom2 == dom3):
```

```
84                 if dom3 not in domainsInD2_Q3:
85                     d2_freq = d2Item["NUM_IN_TWEETS"]
86                     q3_freq = q3Item["NUM_TWEETS"]
87                     tot_freq = d2_freq + q3_freq
88                     new_dict = {"DOMAIN":dom3,"TOTAL_FREQ":tot_freq,"
    D2_FREQ":d2_freq,"Q3_FREQ":q3_freq}
89                     temp2.append(new_dict)
90
91     domainsInD2_Q3 = sorted(temp2, key = itemgetter("TOTAL_FREQ"),
    reverse=True)
92
93 def output(list):
94     for item in list:
95         print(item)
96
97     print("\nNum in list: "+str(len(list)))
98
99 def exportComparisonStats():
100     with open("Q4Stats.txt","w") as f:
101         print("Top 5 Shared Domains in Set D1_Q3:"+"\n")
102         f.write("Top 5 Shared Domains in Set D1_Q3:"+"\n\n")
103         breaker = 0
104         for item in domainsInD1_Q3:
105             if breaker == 5:
106                 break
107             domain = item["DOMAIN"]
108             tot_freq = item["TOTAL_FREQ"]
109             d1_freq = item["D1_FREQ"]
110             q3_freq = item["Q3_FREQ"]
111             print(
112                 "Domain: "+domain+"\n"
113                 "Total Tweet Count from Domain in Set: "+str(tot_freq)+
    "\n"
114                 "Tweet Count from D1: "+str(d1_freq)+"\n"
115                 "Tweet Count from Q3: "+str(q3_freq)+"\n"
116             )
117             f.write(
118                 "Domain: "+domain+"\n"
119                 "Total Tweet Count from Domain in Set: "+str(tot_freq)+
    "\n"
120                 "Tweet Count from D1: "+str(d1_freq)+"\n"
121                 "Tweet Count from Q3: "+str(q3_freq)+"\n\n"
122             )
123             breaker+=1
124
125         #print("\n")
126         #for x in range(20):
```

```
127         print("--"*20)
128         f.write("--"*20)
129 
130         print("\n")
131         f.write("\n\n")
132 
133         print("Top 5 Shared Domains in Set D1_Q3:"+"\n")
134         f.write("Top 5 Shared Domains in Set D1_Q3:"+"\n\n")
135         breaker = 0
136         for item in domainsInD2_Q3:
137             if breaker == 5:
138                 break
139             domain = item["DOMAIN"]
140             tot_freq = item["TOTAL_FREQ"]
141             d2_freq = item["D2_FREQ"]
142             q3_freq = item["Q3_FREQ"]
143             print(
144                 "Domain: "+domain+"\n"
145                 "Total Tweet Count from Domain in Set: "+str(tot_freq)+
    "\n"
146                 "Tweet Count from D2: "+str(d2_freq)+"\n"
147                 "Tweet Count from Q3: "+str(q3_freq)+"\n"
148             )
149             f.write(
150                 "Domain: "+domain+"\n"
151                 "Total Tweet Count from Domain in Set: "+str(tot_freq)+
    "\n"
152                 "Tweet Count from D2: "+str(d2_freq)+"\n"
153                 "Tweet Count from Q3: "+str(q3_freq)+"\n\n"
154             )
155             breaker+=1
156 
157 def drawD1Q3Graph():
158     dataset = []
159     domain =[]
160     frequency = []
161     count=0
162     for item in domainsInD1_Q3:
163         if count == 5:
164             break
165         dom = item["DOMAIN"]
166         d1 = item["D1_FREQ"]
167         q3 = item["Q3_FREQ"]
168 
169         dataset.append("D1")
170         domain.append(dom)
171         frequency.append(d1)
```

```
172
173         dataset.append("Q3")
174         domain.append(dom)
175         frequency.append(q3)
176
177         count+=1
178
179     data_dict = {"Dataset":dataset,"Domain":domain,"Frequency":
    frequency}
180     df = pd.DataFrame(data_dict)
181
182     pal = {"Q3": "darkviolet", "D1": "orangered"}
183
184     sns.catplot(y="Domain", x="Frequency", hue="Dataset", palette=pal,
    kind ="bar",data=df, height=5, aspect=1.5)
185
186     plt.ylabel("Domain")
187     plt.xlabel("Tweet Count")
188     plt.title("Number of Tweets per Domain")
189     plt.show()
190
191 def drawD2Q3Graph():
192     dataset = []
193     domain =[]
194     frequency = []
195     count=0
196     for item in domainsInD2_Q3:
197         if count == 5:
198             break
199         dom = item["DOMAIN"]
200         d2 = item["D2_FREQ"]
201         q3 = item["Q3_FREQ"]
202
203         dataset.append("D2")
204         domain.append(dom)
205         frequency.append(d2)
206
207         dataset.append("Q3")
208         domain.append(dom)
209         frequency.append(q3)
210
211         count+=1
212
213     data_dict = {"Dataset":dataset,"Domain":domain,"Frequency":
    frequency}
214     df = pd.DataFrame(data_dict)
215
```

```
216     pal = {"Q3": "darkviolet", "D2": "gold"}
217
218     sns.catplot(y="Domain", x="Frequency", hue="Dataset", palette=pal,
        kind ="bar",data=df, height=5, aspect=1.5)
219
220     plt.ylabel("Domain")
221     plt.xlabel("Tweet Count")
222     plt.title("Number of Tweets per Domain")
223     plt.show()
224
225
226 if __name__=='__main__':
227     readD1()
228     readD2()
229     readQ3()
230     compareDomains()
231     #output(domainsInD1_Q3)
232     #drawD1Q3Graph()
233     #drawD2Q3Graph()
234     exportComparisonStats()
```

Listing 15: Python code for Q4

Listing 16 below shows the top 5 common domains for each comparison set and the number of times they were featured in each individual data set. These were found by counting the total number of tweets a domain was featured in between each data set.

Based on the results, there were a few domains that shared a top 5 spot in each comparison, namely *dcclothesline.com* and *activistpost.com*. These two had a much larger presence in the D1 data set than the D2 data set. Of note, there is much larger disparity between the tweet counts in the *D1_Q3* distribution than in the *D2_Q3* distribution. *infowars.com* in particular heavily skews the distrbution with its presence in the D1 data.

```
1  Top 5 Shared Domains in Set D1_Q3:
2
3  Domain: infowars.com
4  Total Tweet Count from Domain in Set: 1972
5  Tweet Count from D1: 1741
6  Tweet Count from Q3: 231
7
8  Domain: beforeitsnews.com
9  Total Tweet Count from Domain in Set: 649
10 Tweet Count from D1: 580
11 Tweet Count from Q3: 69
12
13 Domain: dcclothesline.com
14 Total Tweet Count from Domain in Set: 435
15 Tweet Count from D1: 233
```

```
Tweet Count from Q3: 202

Domain: activistpost.com
Total Tweet Count from Domain in Set: 379
Tweet Count from D1: 175
Tweet Count from Q3: 204

Domain: 21stcenturywire.com
Total Tweet Count from Domain in Set: 269
Tweet Count from D1: 62
Tweet Count from Q3: 207

----------------------------------------

Top 5 Shared Domains in Set D2_Q3:

Domain: gellerreport.com
Total Tweet Count from Domain in Set: 438
Tweet Count from D2: 234
Tweet Count from Q3: 204

Domain: dcclothesline.com
Total Tweet Count from Domain in Set: 298
Tweet Count from D2: 96
Tweet Count from Q3: 202

Domain: activistpost.com
Total Tweet Count from Domain in Set: 287
Tweet Count from D2: 83
Tweet Count from Q3: 204

Domain: collective-evolution.com
Total Tweet Count from Domain in Set: 255
Tweet Count from D2: 50
Tweet Count from Q3: 205

Domain: ronpaulinstitute.org
Total Tweet Count from Domain in Set: 250
Tweet Count from D2: 92
Tweet Count from Q3: 158
```

Listing 16: Top 5 domains per data set comparison

Figures 3 and 4 below show visualizations of the above data for each domain in each comparison.

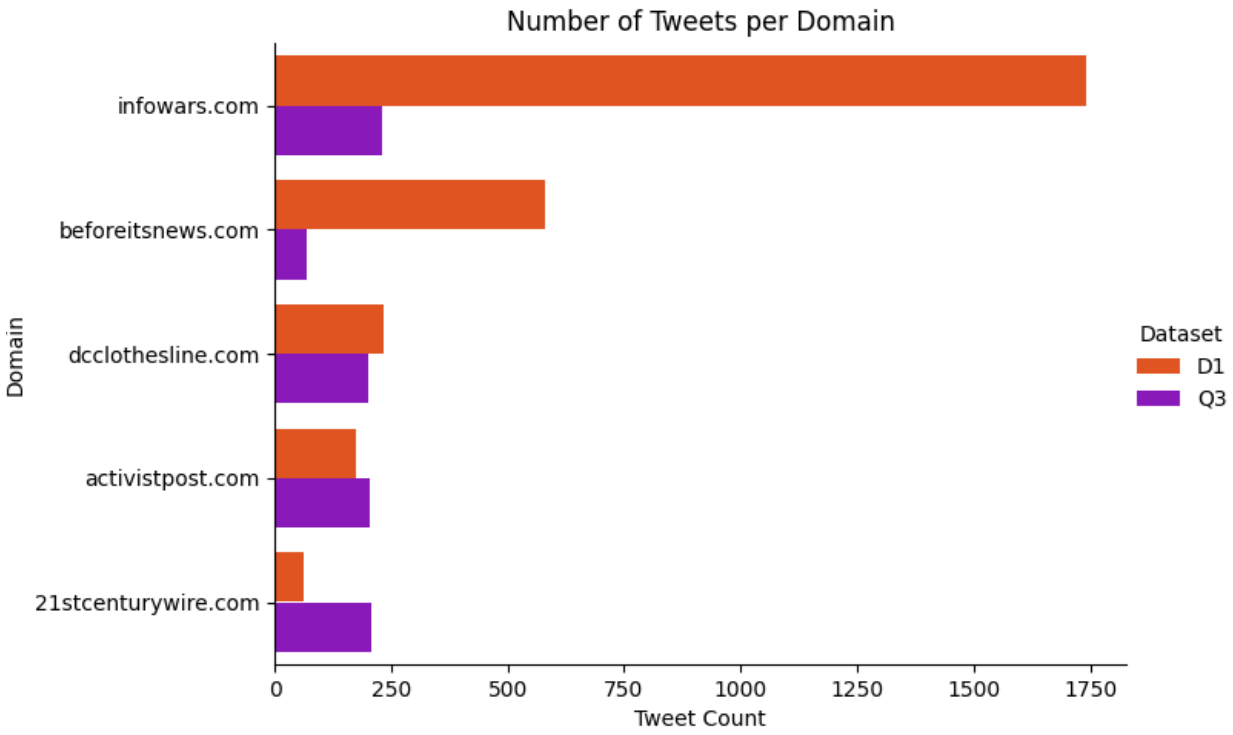

Figure 3: Top 5 domains shared between datasets D1 and Q3

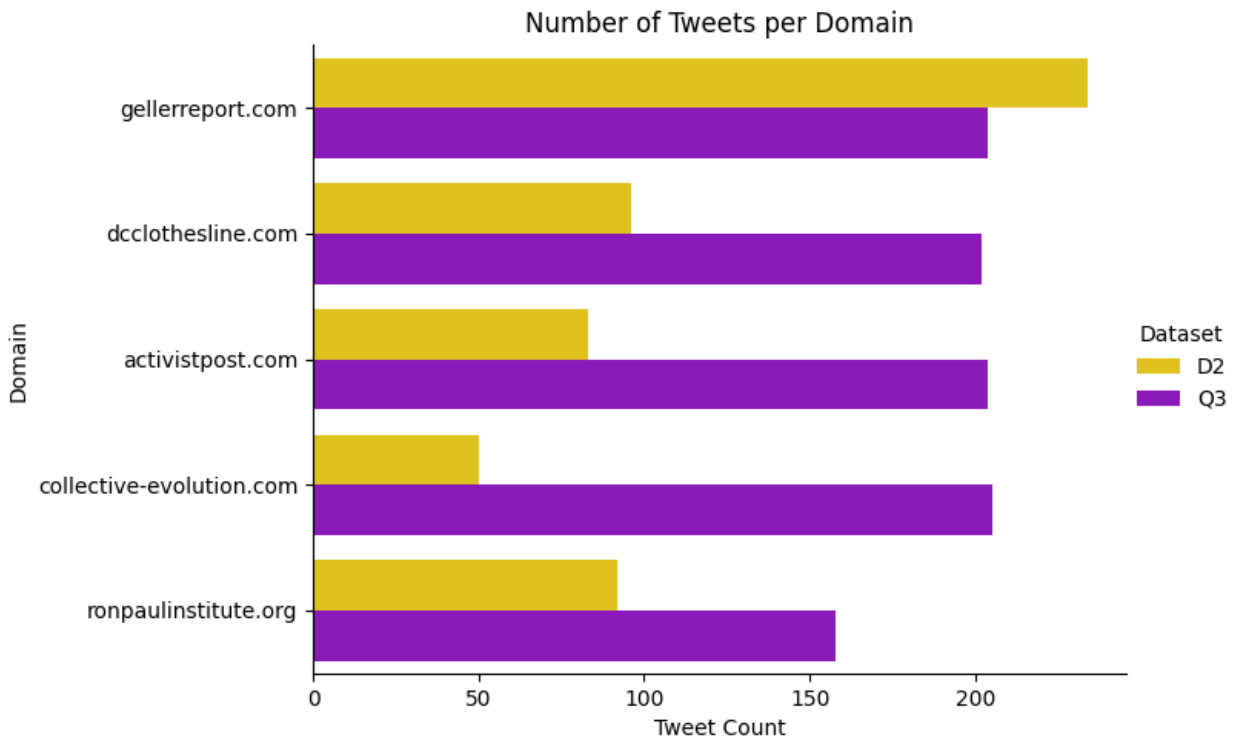

Figure 4: Top 5 domains shared between datasets D2 and Q3

Q6 (Extra Credit)

Listing 17 below features the code used for Q6. In order to properly analyze the text of the collected tweets, I needed to read in *tweets.json* from Q3 and clean each tweet by removing all special characters, URI links, and stopwords from the text bodies to isolate the prominant words in each tweet.

For each tweet, I passed the text into the *removeUrlsAndSpecialChars()* method. This method used a regular expression to remove any non-Alpha characters from the text; the resulting string was also lowercased to help later with removing stopwords. The text was then split into a list to isolate each word, and then stored in another list (*tweet_Bodies*) containing each split text with the same index as the tweet it came from in the original data.

With the resulting list of lists, a nested for-loop was used to access each word in each text body. For each tweet body, a temporary list created, and each word was compared to a list of english stopwords provided by the Natural Language ToolKit (*nltk*) library. If a word was not present in the stopword list (not a stopword), it was added to the temporary list, and the list was stored in a new list called *tweets_no_stops*; this list is identical in structure to the previous (*tweet_Bodies*) minus the stopwords in each text body. To assist with additional analysis, a final string list was created to hold every word across all the cleaned text bodies.

I wanted to find the most common terms across all the gathered tweets to help highlight the overall topics discussed across them. In order to find the most common terms, I used some functions featured in the Counter class from the *collections* library. In the *getMostCommonTerms()* method, I created a Counter object from *all_words* and used *most_common(20)* to get the top 20 most common words and the number of times they appeared (somehow the resulting top two words were stopwords that slipped through cleaning, so I had to get the top 22 words and remove the first two). The words and their counts are returned as individual lists that I stored in *most_common*. I also created a seperate list specifically to hold only the words called *common*.

With the common terms found, I ran a check across all tweets to see which tweets contained any of the common terms; this was handled in the *checkCommonTerms()* method. Going through each split body of text and each common term in a nested loop, a counter was incremented if a common word was present within the text. Since I was only checking if a tweet contained any of the common terms, the boolean *breaker* was used to break from the loop if a common term was found.

```
1 import json
2 from collections import Counter
3 from nltk.corpus import stopwords
4 import re
5 import nltk
6 import itertools
7
8 tweets= []
9 tweet_bodies = []
```

```
10 tweets_no_stops = []
11 all_words = []
12 most_common = []
13 common = []
14 tweets_with_common_terms = 0
15
16 def process():
17     global tweets, tweet_bodies, tweets_no_stops, all_words
18     with open("tweets.json",'r') as f:
19         data = json.load(f)
20         for item in data:
21             tweet_dict = {'ID':item["ID"],'TEXT':item["TEXT_BODY"]}
22             tweets.append(tweet_dict)
23             text_no_url = removeUrlsAndSpecialChars(item["TEXT_BODY"]).
    lower()
24             text_split = text_no_url.split()
25             tweet_bodies.append(text_split)
26
27     #nltk.download('stopwords')
28     stop_words = set(stopwords.words('english'))
29
30     for text in tweet_bodies:
31         temp = []
32         for word in text:
33                 if word not in stop_words:
34                     temp.append(word)
35         tweets_no_stops.append(temp)
36
37     for text in tweets_no_stops:
38         for word in text:
39             all_words.append(word)
40
41 def removeUrlsAndSpecialChars(txt):
42     return " ".join(re.sub(r"([^0-9A-Za-z \t])|(\w+:\/\/\S+)", "", txt)
    .split())
43
44 def getMostCommonTerms():
45     global most_common, common
46     common_words = Counter(all_words)
47
48     most_common = common_words.most_common(22)
49     del most_common[0:2]
50
51     for item in most_common:
52         common.append(item[0])
53
54 def checkCommonTerms():
```

```
55      global tweets_with_common_terms
56      for text in tweet_bodies:
57          for word in common:
58              breaker = False
59              if word in text:
60                  tweets_with_common_terms+=1
61                  breaker = True
62              if breaker:
63                  break
64
65  def output():
66      print("Top 20 most common terms out of "+str(len(tweets))+" Tweets"
        )
67      for word in most_common:
68          print(word)
69      print("\nNumber of Tweets featuring a common term: "+str(
        tweets_with_common_terms))
70
71  if __name__=='__main__':
72      process()
73      getMostCommonTerms()
74      checkCommonTerms()
75      output()
```

Listing 17: Python code for Q6

Listing 18 below shows the common terms, how many times they appeared in the data, and the number of tweets featuring a common term. Of the collected tweets, 1468 featured one of these common terms in their text bodies. Considering the time window of the tweets from Q3, these tweets were posted in the few weeks following the US election. Even without knowing the time window, the large majority of the common terms are all buzzwords used in relation to the US election results, the widespread claims of voter fraud, and the COVID-19 pandemic.

Voter fraud in regards to the 2020 election is a very topical subject, and while these claims have been shot down in every court case due to lack of evidence, misinformation about the election results still continues to circulate and garner a lot of attention. In this context, it makes sense that these terms would be common amongst domains that aim to spread misinformation.

An interesting thing to note, however, is that the actual top term is *election*, as *worldtruthtv* is derived from the domain *worldtruth.tv* (it somehow made it past the link-removal in cleanup). However, I was curious so I looked into it, and it is a literal fake news website; it is even labeled as an ”alternative news network” on the site. It’s clear based on the article content and how they’re worded that the site is pushing out false information based on conspiracy theories. It is slightly disturbing how a site like this and its content can be spread around so easily on social media platforms like Twitter.

```
1 Top 20 most common terms out of 2908 Tweets
2
```

```
('worldtruthtv', 163)
('election', 161)
('trump', 141)
('covid19', 135)
('us', 132)
('state', 128)
('covid', 124)
('cia', 123)
('new', 119)
('people', 113)
('fraud', 111)
('news', 101)
('update', 97)
('war', 94)
('breitbartnews', 93)
('uk', 90)
('dominion', 87)
('says', 87)
('situation', 85)
('vaccine', 84)

Number of Tweets featuring a common term: 1468
```

Listing 18: Most common terms across gathered tweets, ordered by number of tweets they appeared in