Tweet Processor

CS109a: Fall 2018

Authors: Gordon Hew, Wenqin Hu, Blair Leduc

TF: Ken Arnold

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
import tweepy
import sys
import jsonpickle
import os
import zipfile
import time
from datetime import date, datetime, time
import re
import json
from pprint import pprint
import random
from dateutil import tz
from pandas.io.json import json_normalize
import requests
import gzip
import shutil
```

Parameters

In [2]:
```
tweets_file = os.path.join('data','last_100_timeline_tweets.json.gz')
tweets_urls_file = os.path.join('data','tweet_urls.json')

normalized_users_df_file = os.path.join('tmp','users_df.pkl')
normalized_tweets_df_file = os.path.join('tmp','tweets_df.pkl')
normalized_users_df_gz_file = os.path.join('tmp','users_df.pkl.gz')
normalized_tweets_df_gz_file = os.path.join('tmp','tweets_df.pkl.gz')

final_users_df_file = os.path.join('tmp','users_final_df.pkl')
final_tweets_df_file = os.path.join('tmp','tweets_final_df.pkl')
final_users_df_gz_file = os.path.join('tmp','users_final_df.pkl.gz')
final_tweets_df_gz_file = os.path.join('tmp','tweets_final_df.pkl.gz')

botometer_result_1000 = os.path.join('data',
                                     'botometer_result_1000random.json')

run_url_expander = False
```

Extract JSON Tweets into a Pandas Data Frame

In [3]:

```
users_json = list()
tweets_json = list()

with gzip.open(tweets_file, 'rt') as f:
    idx = 0
    for idx, line in enumerate(f):
        line_json = json.loads(line)

        if 'user' in line_json:
            user_json = line_json.pop('user', None)
            users_json.append(user_json)

            line_json['user_id'] = user_json['id']
            line_json['id_str'] = user_json['id_str']
            line_json['screen_name'] = user_json['screen_name']

        tweets_json.append(line_json)

        if ((idx + 1) % 10000 == 0):
            print('Finished Proceessing Tweet #', idx + 1)

    print('Finished Proceessing Tweet #', idx + 1)

users_df = json_normalize(users_json)
tweets_df = json_normalize(tweets_json)

display(users_df.describe())
display(tweets_df.describe())
```

```
Finished Proceessing Tweet # 10000
Finished Proceessing Tweet # 20000
Finished Proceessing Tweet # 30000
Finished Proceessing Tweet # 40000
Finished Proceessing Tweet # 50000
Finished Proceessing Tweet # 60000
Finished Proceessing Tweet # 70000
Finished Proceessing Tweet # 80000
Finished Proceessing Tweet # 90000
Finished Proceessing Tweet # 97854
```

	favourites_count	followers_count	friends_count	id	listed_count	statuses_cour
count	1000.000000	1000.000000	1000.000000	1.000000e+03	1000.000000	1000.00000
mean	21331.004000	1867.186000	1139.991000	3.423497e+17	25.115000	31638.96700
std	39114.780324	7344.047951	4073.343137	4.482004e+17	92.364381	59896.71990
min	0.000000	0.000000	0.000000	3.023431e+06	0.000000	1.00000
25%	1550.250000	108.750000	167.000000	4.943091e+08	0.000000	2309.75000
50%	7982.500000	363.000000	397.500000	2.814042e+09	2.000000	9648.50000
75%	23129.750000	1094.500000	908.250000	8.528221e+17	11.000000	32057.50000
max	383288.000000	100730.000000	85123.000000	1.070908e+18	1550.000000	624250.00000

	coordinates	favorite_count	geo	id	in_reply_to_status_id	in_reply_to_user_id
count	0.0	97854.000000	0.0	9.785400e+04	1.981000e+04	2.087000e+04
mean	NaN	1.532150	NaN	1.062750e+18	1.060179e+18	2.982529e+17
std	NaN	69.065511	NaN	4.524701e+16	5.548095e+16	4.329941e+17
min	NaN	0.000000	NaN	1.681337e+10	1.467126e+10	1.200000e+01
25%	NaN	0.000000	NaN	1.068163e+18	1.067390e+18	2.214333e+08
50%	NaN	0.000000	NaN	1.070557e+18	1.070352e+18	2.167074e+09
75%	NaN	0.000000	NaN	1.071038e+18	1.070917e+18	8.150505e+17
max	NaN	19464.000000	NaN	1.071283e+18	1.071281e+18	1.071065e+18

8 rows × 64 columns

In [4]:
```
# Write as binary file
users_df.to_pickle(normalized_users_df_gz_file, compression = 'gzip')
tweets_df.to_pickle(normalized_tweets_df_gz_file, compression = 'gzip')
```

In [5]:
```
users_pkl_df = pd.read_pickle(normalized_users_df_gz_file,
                              compression = 'gzip')
tweets_pkl_df = pd.read_pickle(normalized_tweets_df_gz_file,
                               compression = 'gzip')
```

In [6]:
```
display(users_pkl_df.describe())
display(users_pkl_df.head())
display(tweets_pkl_df.describe())
display(tweets_pkl_df.head())
```

	favourites_count	followers_count	friends_count	id	listed_count	statuses_cour
count	1000.000000	1000.000000	1000.000000	1.000000e+03	1000.000000	1000.00000
mean	21331.004000	1867.186000	1139.991000	3.423497e+17	25.115000	31638.96700
std	39114.780324	7344.047951	4073.343137	4.482004e+17	92.364381	59896.71990
min	0.000000	0.000000	0.000000	3.023431e+06	0.000000	1.00000
25%	1550.250000	108.750000	167.000000	4.943091e+08	0.000000	2309.75000
50%	7982.500000	363.000000	397.500000	2.814042e+09	2.000000	9648.50000
75%	23129.750000	1094.500000	908.250000	8.528221e+17	11.000000	32057.50000
max	383288.000000	100730.000000	85123.000000	1.070908e+18	1550.000000	624250.00000

	contributors_enabled	created_at	default_profile	default_profile_image	description	entities.d
0	False	Sun Nov 26 00:14:54 +0000 2017	True	False	Unapologetic advocate of common law, constitut...	'https://t.co
1	False	Fri Apr 03 02:54:56 +0000 2015	True	False	definitely a real human woman and not three du...	
2	False	Wed Apr 15 00:56:50 +0000 2015	True	False	where i end, you'll begin.	
3	False	Sat Oct 20 20:11:16 +0000 2012	False	False	it's Kah-doom • UNC • Nigerian • it's not by f...	
4	False	Fri Mar 30 02:47:09 +0000 2012	False	False	21.	

5 rows × 43 columns

	coordinates	favorite_count	geo	id	in_reply_to_status_id	in_reply_to_user_id
count	0.0	97854.000000	0.0	9.785400e+04	1.981000e+04	2.087000e+04
mean	NaN	1.532150	NaN	1.062750e+18	1.060179e+18	2.982529e+17
std	NaN	69.065511	NaN	4.524701e+16	5.548095e+16	4.329941e+17
min	NaN	0.000000	NaN	1.681337e+10	1.467126e+10	1.200000e+01
25%	NaN	0.000000	NaN	1.068163e+18	1.067390e+18	2.214333e+08
50%	NaN	0.000000	NaN	1.070557e+18	1.070352e+18	2.167074e+09
75%	NaN	0.000000	NaN	1.071038e+18	1.070917e+18	8.150505e+17
max	NaN	19464.000000	NaN	1.071283e+18	1.071281e+18	1.071065e+18

8 rows × 64 columns

	contributors	coordinates	coordinates.coordinates	coordinates.type	created_at	entities.hashta
0	None	NaN	NaN	NaN	Sat Dec 08 01:50:37 +0000 2018	
1	None	NaN	NaN	NaN	Sat Dec 08 01:50:18 +0000 2018	[{'tex 'Pelosi201 'indices': [6, 17
2	None	NaN	NaN	NaN	Sat Dec 08 01:46:30 +0000 2018	[{'tex 'BigBard 'indices': [3 46]}, {'
3	None	NaN	NaN	NaN	Sat Dec 08 01:45:48 +0000 2018	[{'tex 'ScottFre 'indices': [9 101
4	None	NaN	NaN	NaN	Sat Dec 08 01:45:28 +0000 2018	[{'tex 'ScottFre 'indices': [4 57

5 rows × 318 columns

In [7]:
```
len(tweets_pkl_df.index)
```

Out[7]: 97854

Website Classification Processing

Extract Website URLs From Tweets

In [8]:
```
tweets_pkl_df.head()
```

Out[8]:

	contributors	coordinates	coordinates.coordinates	coordinates.type	created_at	entities.hashta
0	None	NaN	NaN	NaN	Sat Dec 08 01:50:37 +0000 2018	
1	None	NaN	NaN	NaN	Sat Dec 08 01:50:18 +0000 2018	[{'tex 'Pelosi201 'indices': [6, 17
2	None	NaN	NaN	NaN	Sat Dec 08 01:46:30 +0000 2018	[{'tex 'BigBard 'indices': [3 46]}, {'
3	None	NaN	NaN	NaN	Sat Dec 08 01:45:48 +0000 2018	[{'tex 'ScottFre 'indices': [9 101
4	None	NaN	NaN	NaN	Sat Dec 08 01:45:28 +0000 2018	[{'tex 'ScottFre 'indices': [4 57

5 rows × 318 columns

In [9]:
```
tweets_with_urls_df = tweets_pkl_df[tweets_pkl_df['entities.urls']\
                                    .map(lambda urls: len(urls)) > 0]

print('tweets with links:', len(tweets_with_urls_df.index))
display(tweets_with_urls_df.head())
display(tweets_with_urls_df.describe())
```

```
tweets with links: 23415
```

	contributors	coordinates	coordinates.coordinates	coordinates.type	created_at	entities.h
1	None	NaN	NaN	NaN	Sat Dec 08 01:50:18 +0000 2018	[{'text': 'Pel 'indices': [6
8	None	NaN	NaN	NaN	Sat Dec 08 01:44:41 +0000 2018	[{'text': 'Sc 'indices':
11	None	NaN	NaN	NaN	Sat Dec 08 01:42:15 +0000 2018	'DemoteTrump 'indice
14	None	NaN	NaN	NaN	Sat Dec 08 01:40:19 +0000 2018	[{'text': 'in 'indices': [0
15	None	NaN	NaN	NaN	Sat Dec 08 01:39:53 +0000 2018	

5 rows × 318 columns

	coordinates	favorite_count	geo	id	in_reply_to_status_id	in_reply_to_user_id
count	0.0	23415.000000	0.0	2.341500e+04	3.396000e+03	3.966000e+03
mean	NaN	1.900577	NaN	1.062437e+18	1.057405e+18	2.688713e+17
std	NaN	34.667200	NaN	3.544404e+16	5.865803e+16	4.192413e+17
min	NaN	0.000000	NaN	2.936052e+17	3.250720e+17	3.599300e+04
25%	NaN	0.000000	NaN	1.067670e+18	1.066981e+18	1.135979e+08
50%	NaN	0.000000	NaN	1.070486e+18	1.070216e+18	9.507473e+08
75%	NaN	1.000000	NaN	1.071018e+18	1.070919e+18	7.844378e+17
max	NaN	3521.000000	NaN	1.071283e+18	1.071278e+18	1.071065e+18

8 rows × 64 columns

In [10]:
```
if run_url_expander:
    session = requests.Session()

    tweet_urls = list()

    for i,(tweet_idx, row) in enumerate(tweets_with_urls_df.iterrows()):
        # visit the URL and get the expanded site
        for url_idx, url in enumerate(row['entities.urls']):
            try:
                resp = session.head(url['expanded_url'],
                                    allow_redirects = True)
                tweet_urls.append({'expanded_url': url['expanded_url'],
                                   'unshortened_url': resp.url })
            except Exception as err:
                print('Unable to process url', url['expanded_url'], err)

        if ((i + 1) % 100 == 0):
            print('Finished Proceessing Tweet #', i + 1)

    print('Finished Proceessing Tweet #', i + 1)

    with open('tweet_urls.json', 'w') as f:
        for url_json in tweet_urls:
            f.write(json.dumps(url_json))
            f.write('\n')

    print('Processed', len(tweet_urls), 'URLs Successfully')
```

Classify Websites Using Rules

In [11]:
```
def categorize_website(url):

    is_twitter = url.startswith('https://twitter.com') \
        or url.startswith('https://mobile.twitter.com')

    if is_twitter:
        return 'Twitter'

    other_social_media_urls = (
        'https://i.imgur.com',
        'https://www.instagram.com',
        'https://www.facebook.com',
        'https://m.soundcloud.com',
        'https://soundcloud.com',
        'https://www.linkedin.com',
        'https://vine.co',
        'http://www.facebook.com',
    )

    if url.startswith(other_social_media_urls):
        return 'Other Social Media'

    digital_media_urls = (
        'https://www.twitch.tv',
        'https://www.youtube.com',
        'https://open.spotify.com/',
        'http://www.youtube.com',
        'http://mpg.dnset.com',
        'https://www.pscp.tv',
    )

    if url.startswith(digital_media_urls):
        return 'Digital Media'

    news_urls = (
        'https://www.nytimes.com',
        'https://www.newsweek.com',
        'https://abc7.com',
        'https://news.sky.com',
        'https://www.bbc.com',
        'https://www.sbs.com.au',
        'https://www.yahoo.com/news/',
        'https://www.yahoo.com/lifestyle/',
        'https://abcnews.go.com',
        'https://www.wsj.com',
        'https://thehill.com',
        'https://www.foxnews.com',
        'https://www.usatoday.com',
        'https://www.sun-sentinel.com',
        'https://www.washingtontimes.com',
        'https://www.washingtonpost.com',
        'http://time.com',
        'https://fox8.com',
        'https://www.huffingtonpost.com',
        'http://asiannewsservice.in',
        'http://austin.culturemap.com',
```

```
        'https://www.reuters.com',

        # business
        'https://www.forbes.com',
        'https://www.zerohedge.com',
        'https://thebrandboy.com',

        # Gaming
        'https://www.gameinformer.com',
        'https://mynintendonews.com',

        # Tech
        'https://www.engadget.com',
        'https://arstechnica.com',
        'https://www.zdnet.com',

        'https://www.travelandleisure.com',
        'http://dallas.culturemap.com',

        'https://www.statesman.com',

        'https://www.washingtonexaminer.com',
        'https://www.dailywire.com',
        'https://www.dailymail.co.uk',
        'http://www.dailymail.co.uk',
        'https://www.theblaze.com',
        'https://www.breitbart.com',
        'https://dailycaller.com',
        'https://paper.li',
        'http://www.kohimanewspaper.org',
        'http://geteducation.com.au',
        'https://morgan-magazine.com',
        'http://www.travelweekly.co.uk',
        'https://apple.news',
        'http://getstem.com.au',
        'https://www.bloomberg.com',
        'http://thefederalist.com',
        'http://getindustry.com.au',
        'https://www.thedailybeast.com',
        'https://www.hannity.com',
        'https://www.cnbc.com',
        'https://hillreporter.com',
        'https://www.politicususa.com',
        'https://www.mesopinions.com',
        'https://www.theguardian.com',
        'https://www.cpr.org',
        'https://www.cnn.com',
        'https://www.thegatewaypundit.com',
    )

    if url.startswith(news_urls):
        return 'News and Current Events'

    commercial_product = (
        'https://store.playstation.com',
        'https://www.starz.com',
        'https://www.amazon.com',
```

```
        'https://represent.com/store/pewdiepie',
        'https://itunes.apple.com',
        'https://sumall.com',
        'http://fllwrs.com/',
        'http://www.tweematic.com',
        'https://www.bible.com/', # sells bible apps
        'https://www.xbox.com',
        'https://www1.ticketmaster.com',
        'https://www.zazzle.com',
        'https://www.woodenrings.com',
        'https://www.workable.com',
        'http://www.twitlonger.com',

        'https://www.twittascope.com',
        'https://pages.ebay.com',
        'https://www.ebay.co.uk',
        'https://curiouscat.me',
        'https://api.curations.bazaarvoice.com',
        'https://www.exclusivetravelrates.com',
        'https://poshmark.com',
        'http://foodtronic.com',
        'https://www.etsy.com',
        'https://www.commissioncut.ca',
        'http://www.edumine.com',
        'http://smarturl.it',
        'http://naver.me',
        'https://www.gofundme.com',
        'https://www.documentcloud.org',
        'https://mailchi.mp',
        'https://fanlink.to',
        'http://nocturnalvegas.com',
        'https://www.bandsintown.com',
        'https://trakt.tv',
        'https://www.crowdfireapp.com',
        'https://gifkaro.app.link',
        'https://empire.lnk.to',
        'https://www.careerwebsite.com',
        'http://epphany.com',
    )

    if url.startswith(commercial_product):
        return 'Commercial Product & Services'

    celebrities_urls = (
        'https://people.com',
        'https://www.hollywoodreporter.com',
        'https://www.allkpop.com',
        'https://www.tmz.com',
        'https://ew.com',
        'https://www.out.com',
        'https://www.soompi.com',
        'https://www.billboard.com',
        'https://www.telltaletv.com',
        'https://entertain.naver.com',
        'https://onlyfans.com',
        'https://starcinema.abs-cbn.com',
        'https://bongino.com',
```

```
        'http://exo.smtown.com',
        'https://www.complex.com',
    )

    if url.startswith(celebrities_urls):
        return 'Celebrities'

    organizations_urls = (
        'https://www.nasa.gov',
        'https://www.justice.gov',
        'http://www.pewresearch.org',
        'https://nca2018.globalchange.gov',
        'http://cc.com/vote',
        'https://www.patriotguard.org',
        'https://www.weforum.org',
        'https://forcechange.com',
        'https://animalpetitions.org',
    )

    if url.startswith(organizations_urls):
        return 'Organization'

    sports_urls = (
        'https://www.clevelandbrowns.com',
        'https://www.sbnation.com',
        'https://mlb.nbcsports.com',
        'https://silverandblacktoday.com',
        'https://www.mlb.com',
        'http://www.lpga.com',
        'https://www.wwe.com',
        'https://www.lpga.or.jp',
        'https://www.nhl.com',
    )

    if url.startswith(sports_urls):
        return 'Sports'

    adult_urls = (
        'https://justfor.fans',
        'https://id10893.weebly.com',
        'https://www.manyvids.com',
    )

    if url.startswith(adult_urls):
        return 'Adult Content'

    unknown_urls = (
        'https://t1.daumcdn.net',
        'https://t.hrtye.com',
    )

    if url.startswith(unknown_urls):
        return 'Unknown'

    return 'Uncategorized'
```

In [12]:

```
pd.set_option('display.max_colwidth', -1)
pd.set_option('display.max_rows', 100)

urls = list()

with open(tweets_urls_file, 'r') as f:
    for url_json in f:
        url_json = json.loads(url_json)

        url_json['category'] \
            = categorize_website(url_json['unshortened_url'])

        unshortened_url = url_json['unshortened_url']

        if unshortened_url.startswith('https://'):
            begin_index = len('https://')
        else:
            begin_index = len('http://')

        try:
            index = unshortened_url.index('/', begin_index)
            url_json['domain'] = unshortened_url[:index]
        except:
            url_json['domain'] = unshortened_url

        urls.append(url_json)

urls_df = pd.DataFrame(urls)

grouped = urls_df.groupby(['domain', 'category'])

top_100_websites_df = pd.DataFrame({'count':urls_df\
                                    .groupby(["domain", "category"])\
                                    .size()})\
                                    .nlargest(100, 'count')\
                                    .reset_index()
display(top_100_websites_df)
```

	domain	category	count
0	https://twitter.com	Twitter	16484
1	https://www.youtube.com	Digital Media	989
2	https://www.instagram.com	Other Social Media	313
3	https://www.facebook.com	Other Social Media	245
4	https://paper.li	News and Current Events	207
5	https://www.twittascope.com	Commercial Product & Services	161
6	https://pages.ebay.com	Commercial Product & Services	120
7	https://soundcloud.com	Other Social Media	107
8	https://thebrandboy.com	News and Current Events	100
9	https://www.ebay.co.uk	Commercial Product & Services	100
10	https://curiouscat.me	Commercial Product & Services	97
11	http://mpg.dnset.com	Digital Media	96
12	https://mailchi.mp	Commercial Product & Services	94
13	http://asiannewsservice.in	News and Current Events	93
14	https://api.curations.bazaarvoice.com	Commercial Product & Services	91
15	https://www.exclusivetravelrates.com	Commercial Product & Services	90
16	http://foodtronic.com	Commercial Product & Services	86
17	https://poshmark.com	Commercial Product & Services	82
18	https://www.twitch.tv	Digital Media	81
19	https://t.hrtye.com	Unknown	78
20	https://gifkaro.app.link	Commercial Product & Services	77
21	https://itunes.apple.com	Commercial Product & Services	74
22	http://fllwrs.com	Commercial Product & Services	73
23	https://www.careerwebsite.com	Commercial Product & Services	71
24	https://open.spotify.com	Digital Media	65
25	https://www.bible.com	Commercial Product & Services	63
26	http://www.kohimanewspaper.org	News and Current Events	60
27	http://geteducation.com.au	News and Current Events	55
28	https://www.patriotguard.org	Organization	51
29	https://morgan-magazine.com	News and Current Events	48
30	https://www.soompi.com	Celebrities	48
31	https://www.weforum.org	Organization	45
32	http://www.travelweekly.co.uk	News and Current Events	42
33	https://www.linkedin.com	Other Social Media	34

	domain	category	count
34	https://thehill.com	News and Current Events	33
35	https://www.pscp.tv	Digital Media	31
36	https://www.nasa.gov	Organization	30
37	https://t1.daumcdn.net	Unknown	28
38	https://www.washingtonpost.com	News and Current Events	28
39	https://www.billboard.com	Celebrities	27
40	https://www.breitbart.com	News and Current Events	27
41	https://www.etsy.com	Commercial Product & Services	27
42	https://www.telltaletv.com	Celebrities	27
43	http://time.com	News and Current Events	25
44	https://entertain.naver.com	Celebrities	25
45	http://epphany.com	Commercial Product & Services	23
46	https://onlyfans.com	Celebrities	23
47	https://www.engadget.com	News and Current Events	22
48	https://www.commissioncut.ca	Commercial Product & Services	21
49	http://www.edumine.com	Commercial Product & Services	20
50	https://apple.news	News and Current Events	20
51	http://getstem.com.au	News and Current Events	18
52	https://www.foxnews.com	News and Current Events	18
53	https://www.nytimes.com	News and Current Events	18
54	https://www.usatoday.com	News and Current Events	18
55	http://smarturl.it	Commercial Product & Services	17
56	https://starcinema.abs-cbn.com	Celebrities	17
57	https://sumall.com	Commercial Product & Services	17
58	https://www.cpr.org	News and Current Events	17
59	https://www.forbes.com	News and Current Events	17
60	https://www.bbc.com	News and Current Events	16
61	https://www.reuters.com	News and Current Events	16
62	http://naver.me	Commercial Product & Services	15
63	https://store.playstation.com	Commercial Product & Services	15
64	https://www.bloomberg.com	News and Current Events	14
65	https://www.cnn.com	News and Current Events	14
66	https://id10893.weebly.com	Adult Content	13
67	https://vine.co	Other Social Media	13
68	https://www.crowdfireapp.com	Commercial Product & Services	13

	domain	category	count
69	https://www.huffingtonpost.com	News and Current Events	13
70	https://www.manyvids.com	Adult Content	13
71	https://www.thegatewaypundit.com	News and Current Events	13
72	http://exo.smtown.com	Celebrities	12
73	https://animalpetitions.org	Organization	12
74	https://empire.lnk.to	Commercial Product & Services	12
75	https://www.amazon.com	Commercial Product & Services	12
76	https://www.complex.com	Celebrities	12
77	https://www.theguardian.com	News and Current Events	12
78	https://dailycaller.com	News and Current Events	11
79	https://trakt.tv	Commercial Product & Services	11
80	https://www.bandsintown.com	Commercial Product & Services	11
81	https://www.lpga.or.jp	Sports	11
82	https://www.mesopinions.com	News and Current Events	11
83	https://www.politicususa.com	News and Current Events	11
84	https://www.zerohedge.com	News and Current Events	11
85	http://nocturnalvegas.com	Commercial Product & Services	10
86	https://bongino.com	Celebrities	10
87	https://fanlink.to	Commercial Product & Services	10
88	https://forcechange.com	Organization	10
89	https://hillreporter.com	News and Current Events	10
90	https://justfor.fans	Adult Content	10
91	https://people.com	Celebrities	10
92	https://www.cnbc.com	News and Current Events	10
93	https://www.documentcloud.org	Commercial Product & Services	10
94	https://www.gofundme.com	Commercial Product & Services	10
95	https://www.hannity.com	News and Current Events	10
96	https://www.thedailybeast.com	News and Current Events	10
97	http://getindustry.com.au	News and Current Events	9
98	http://thefederalist.com	News and Current Events	9
99	http://www.facebook.com	Other Social Media	9

In [13]:
```
#urls_df[['expanded_url', 'unshortened_url']].to_dict('series')
urls_dict = pd.Series(urls_df.unshortened_url.values,
                      index = urls_df.expanded_url).to_dict()
```

In [1]:

```
tweets_url_cat_df = tweets_pkl_df.copy(deep = True)

tweets_url_cat_df['links_to_twitter'] = 0
tweets_url_cat_df['links_to_top_social_media'] = 0
tweets_url_cat_df['links_to_top_digital_media'] = 0
tweets_url_cat_df['links_to_top_news'] = 0
tweets_url_cat_df['links_to_top_products_services'] = 0
tweets_url_cat_df['links_to_top_celebrities'] = 0
tweets_url_cat_df['links_to_top_organizations'] = 0
tweets_url_cat_df['links_to_top_sports'] = 0
tweets_url_cat_df['links_to_top_adult'] = 0

for i, (tweet_idx, row) in enumerate(tweets_url_cat_df.iterrows()):
    # visit the URL and get the expanded site

    if 'entities.urls' in row and len(row['entities.urls']) > 0:

        for url_idx, url in enumerate(row['entities.urls']):

            if url['expanded_url'] in urls_dict:

                unshortened_url = urls_dict[url['expanded_url']]
                category = categorize_website(unshortened_url)

                if category == 'Twitter':
                    row['links_to_twitter'] = row['links_to_twitter']+1
                    tweets_url_cat_df.loc[tweet_idx,
                                          'links_to_twitter'] \
                                            = row['links_to_twitter']
                elif category == 'Other Social Media':
                    row['links_to_top_social_media'] \
                            = row['links_to_top_social_media'] + 1
                    tweets_url_cat_df.loc[tweet_idx,
                                'links_to_top_social_media'] \
                                    = row['links_to_top_social_media']
                elif category == 'Digital Media':
                    row['links_to_top_digital_media'] \
                        = row['links_to_top_digital_media'] + 1
                    tweets_url_cat_df.loc[tweet_idx,
                                'links_to_top_digital_media'] \
                                    = row['links_to_top_digital_media']

                elif category == 'News and Current Events':
                    row['links_to_top_news']=row['links_to_top_news']+1
                    tweets_url_cat_df.loc[tweet_idx,
                                          'links_to_top_news']
                                            = row['links_to_top_news']

                elif category == 'Commercial Product & Services':
                    row['links_to_top_products_services'] \
                        = row['links_to_top_products_services'] + 1
                    tweets_url_cat_df.loc[tweet_idx,
                            'links_to_top_products_services'] \
                                = row['links_to_top_products_services']

                elif category == 'Celebrities':
```

```
                    row['links_to_top_celebrities'] \
                        = row['links_to_top_celebrities'] + 1
                    tweets_url_cat_df.loc[tweet_idx,
                                'links_to_top_celebrities']
                                    = row['links_to_top_celebrities']
                elif category == 'Organization':
                    row['links_to_top_organizations'] \
                        = row['links_to_top_organizations'] + 1
                    tweets_url_cat_df.loc[tweet_idx,
                                'links_to_top_organizations'] \
                                    = row['links_to_top_organizations']

                elif category == 'Sports':
                    row['links_to_top_sports'] \
                        = row['links_to_top_sports'] + 1
                    tweets_url_cat_df.loc[tweet_idx,
                                        'links_to_top_sports'] \
                                            = row['links_to_top_sports']

                elif category == 'Adult Content':
                    row['links_to_top_adult'] \
                        = row['links_to_top_adult'] + 1
                    tweets_url_cat_df.loc[tweet_idx,
                                    'links_to_top_adult'] \
                                            = row['links_to_top_adult']
```

Append Bot-o-meter Score to User Accounts

In [15]:
```
with open(botometer_result_1000, 'rt') as bot_read:
    botometer_result = json.load(bot_read)

botometer_results_df = json_normalize(botometer_result)
users_final_df = users_pkl_df.copy(deep = True)
botometer_results_df = botometer_results_df[['user.id_str',
                                             'scores.universal']]
users_final_df = users_final_df.merge(botometer_results_df, left_on='id_
str',
                                      right_on='user.id_str', how = 'lef
t')
```

Append Bot-o-meter Score to Tweets

In [16]:
```
tweets_final_df = tweets_url_cat_df.copy(deep = True)
botometer_results_df = botometer_results_df[['user.id_str',
                                             'scores.universal']]
tweets_final_df = tweets_final_df.merge(botometer_results_df, left_on='i
d_str',
                                        right_on='user.id_str', how = 'l
eft')
```

Write out Final Data Frames for Aggregation Processing

In [17]:
```
users_final_df.to_pickle(final_users_df_gz_file, compression = 'gzip')
tweets_final_df.to_pickle(final_tweets_df_gz_file, compression = 'gzip')
```

In [ ]:
```

```