In [1]:

```
#!pip install wordcloud
#!pip install textblob
```

In [3]:

```
from PIL import Image
import os
import pandas as pd
import numpy as np
from collections import Counter
import matplotlib.pyplot as plt
import matplotlib
import regex as re

%matplotlib inline

from textblob import TextBlob
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator

import warnings
warnings.filterwarnings('ignore')
```

Let's do some exploratory data analysis and visualization! First let's load our raw data: tweets by Donald Trump.

In [4]:

```
data = pd.read_csv("../data/tweets_01-08-2021.csv")
print(data.shape)
```

```
(56571, 9)
```

We have ~56000 rows and 9 features. Let's take a look at some rows in our dataframe

In [5]:

```
data.head(10)
```

Out[5]:

	id	text	isRetweet	isDeleted	device	favorites	retweets	date	isFlagged
0	98454970654916608	Republicans and Democrats have both created ou...	f	f	TweetDeck	49	255	2011-08-02 18:07:48	f
1	1234653427789070336	I was thrilled to be back in the Great city of...	f	f	Twitter for iPhone	73748	17404	2020-03-03 01:34:50	f

	id	text	isRetweet	isDeleted	device	favorites	retweets	date	isFlagged
2	1218010753434820614	RT @CBS_Herridge: READ: Letter to surveillance...	t	f	Twitter for iPhone	0	7396	2020-01-17 03:22:47	f
3	1304875170860015617	The Unsolicited Mail In Ballot Scam is a major...	f	f	Twitter for iPhone	80527	23502	2020-09-12 20:10:58	f
4	1218159531554897920	RT @MZHemingway: Very friendly telling of even...	t	f	Twitter for iPhone	0	9081	2020-01-17 13:13:59	f
5	1217962723234983937	RT @WhiteHouse: President @realDonaldTrump ann...	t	f	Twitter for iPhone	0	25048	2020-01-17 00:11:56	f
6	1223640662689689602	Getting a little exercise this morning! https:...	f	f	Twitter for iPhone	285863	30209	2020-02-01 16:14:02	f
7	1319501865625784320	https://t.co/4qwCKQOiOw	f	f	Twitter for iPhone	130822	19127	2020-10-23 04:52:14	f
8	1319500520126664705	https://t.co/VlEu8yyovv	f	f	Twitter for iPhone	153446	20275	2020-10-23 04:46:53	f
9	1319500501269041154	https://t.co/z5CRqHO8vg	f	f	Twitter for iPhone	102150	14815	2020-10-23 04:46:49	f

Interesting, we have quite a lot of information on his tweets. We know the date, device, how many favorites and retweets it received, whether it was a retweet itself, and whether it was deleted, which is our outcome feature in this project. For the text itself, we can see that some tweets have the plain text itself, while others have "RT" text and links cluttering the text. We'll have to handle that as we analyze the text closely.

Before we visualize our features, can we get an idea of what these tweets look like? Can we look at the most popular tweets and see what they say?

In [6]:

```
df_favorite = data.sort_values(by = ['favorites'], ascending = False)
N = 5
top_N = df_favorite.iloc[0:N, 1]
for i in range(0,N):
    print(str(i + 1) + ". " + top_N.iloc[i])
    print()
```

```
1. Tonight, @FLOTUS and I tested positive for COVID-19. We will begin our quarantine and recovery process immediatel
y. We will get through this TOGETHER!
```

```
2. Going welI, I think! Thank you to all. LOVE!!!

3. I WON THIS ELECTION, BY A LOT!

4. WE WILL WIN!

5. 71,000,000 Legal Votes. The most EVER for a sitting President!
```

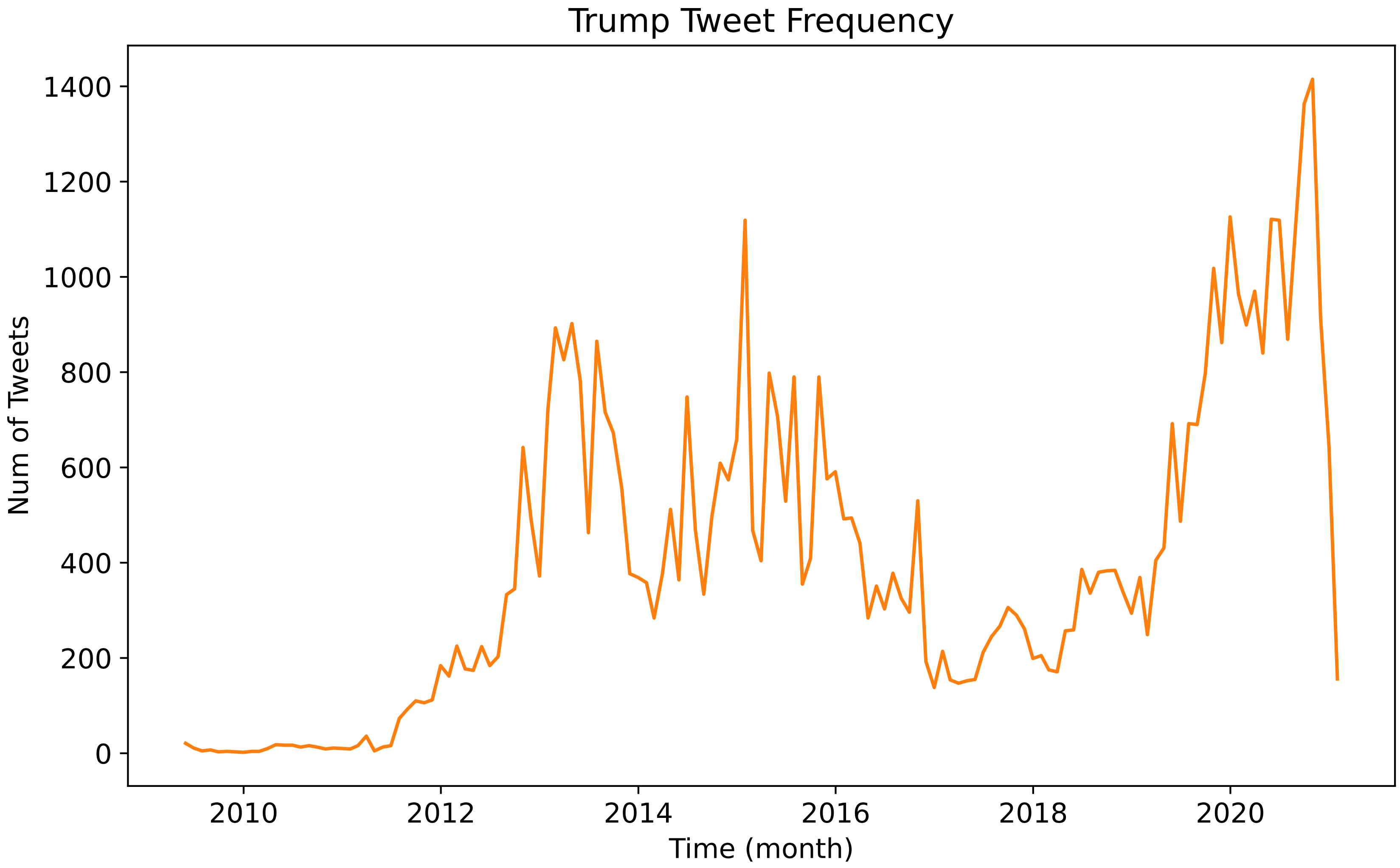
We can see that we have plain text, with some numbers as well as "@" signs. Also, it appears that his most favorited tweets are related to his Covid positivity and his false 2020 election claims.

Now, let's dive into some visualization...

Let's graph the frequency of tweets over time, to get an idea of our spread over time.

In [7]:

```
# Group tweets by month
data['date']=pd.to_datetime(data['date'])
date_group = data
date_group.index = pd.to_datetime(data['date'])
date_group = data.groupby(pd.Grouper(freq='M')).count()
date_group = date_group["id"]

# Plot frequency graph
plt.rc('font', size=12)
fig, ax = plt.subplots(figsize=(10, 6))

ax.plot(date_group.index, date_group, color='tab:orange')

ax.set_xlabel('Time (month)')
ax.set_ylabel('Num of Tweets')
ax.set_title('Trump Tweet Frequency')
plt.savefig("../figs/tweet_frequency.jpg")
```



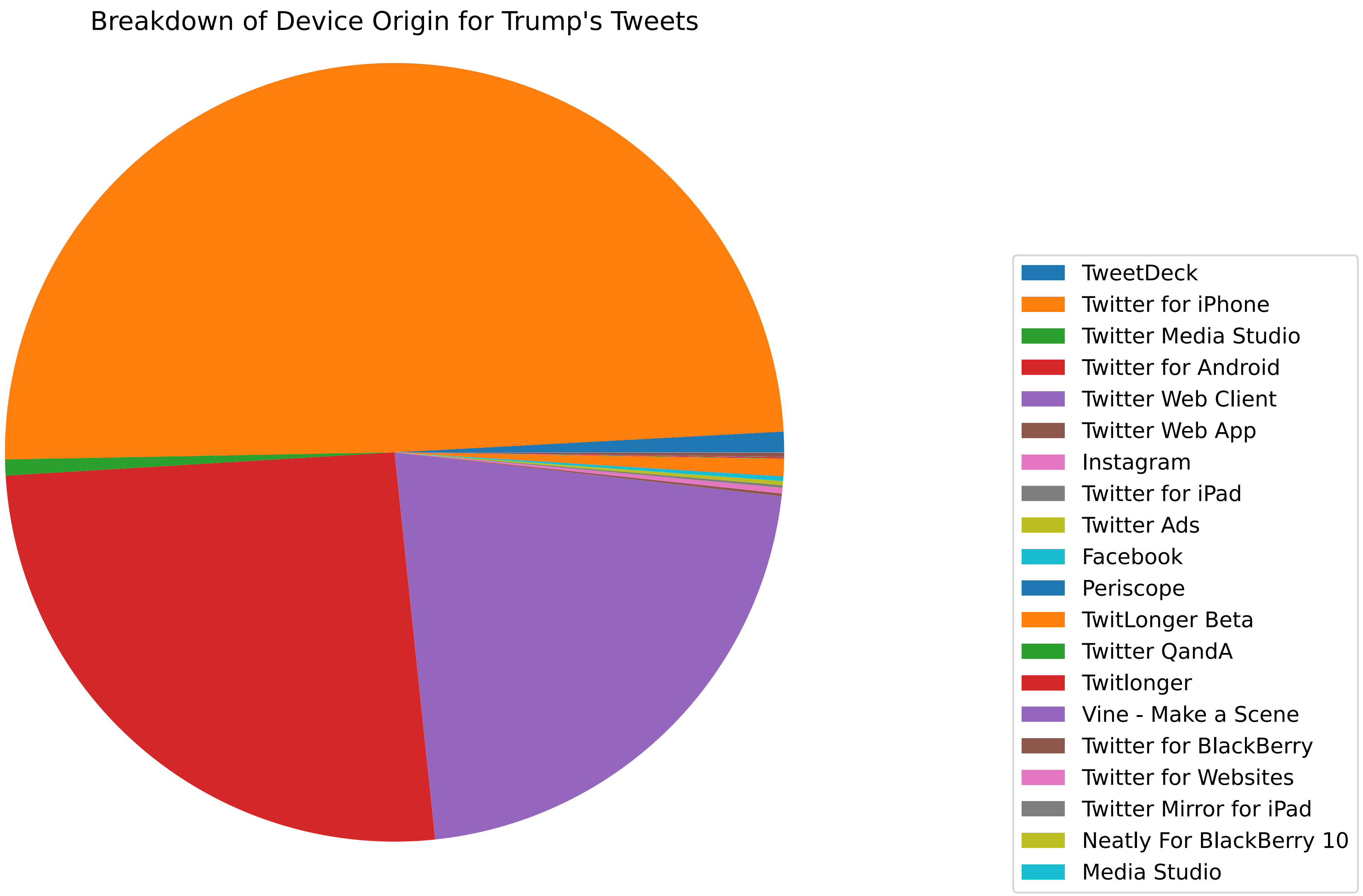
There are some expected and unexpected results from this graph. Expectedly, we see that Trump's tweet frequency follows US election cycles, picking up 2015-2016 and 2019-2020. What is slightly unexpected is how sharply the decline in tweets was after each election, and how many more tweets there were in the 202 election cycle as opposed to the 2016 election cycle.

Now, let's see if we can chart the breakdown of which devices the tweets come from, to see if they're all coming from Trump's personal phone, or from several devices.

In [8]:

```
# Pie chart for devices
n = data.shape[0]
d = data.shape[1]

device_list = data['device'].unique()
device_ratios = []
for device in device_list:
    curr_ratio = data[data['device'] == device].shape[0]/n
    device_ratios.append(curr_ratio)

#myexplode = [0.2, 0, 0, 0]
print(device_list)
fig1, ax1 = plt.subplots(figsize=(6, 5))

ax1.pie(device_ratios, radius=2)
ax1.legend(
    loc='center right',
    labels = device_list,
    prop={'size': 12},
    bbox_to_anchor=(-0.5, 0, 3, 0.5),
)
ax1.set_title("Breakdown of Device Origin for Trump's Tweets", pad = 100)
plt.show()
fig1.savefig("../figs/device_chart.jpg", bbox_inches = 'tight')
```

```
['TweetDeck' 'Twitter for iPhone' 'Twitter Media Studio'
 'Twitter for Android' 'Twitter Web Client' 'Twitter Web App' 'Instagram'
 'Twitter for iPad' 'Twitter Ads' 'Facebook' 'Periscope' 'TwitLonger Beta'
 'Twitter QandA' 'Twitlonger' 'Vine - Make a Scene'
 'Twitter for BlackBerry' 'Twitter for Websites' 'Twitter Mirror for iPad'
 'Neatly For BlackBerry 10' 'Media Studio']
```



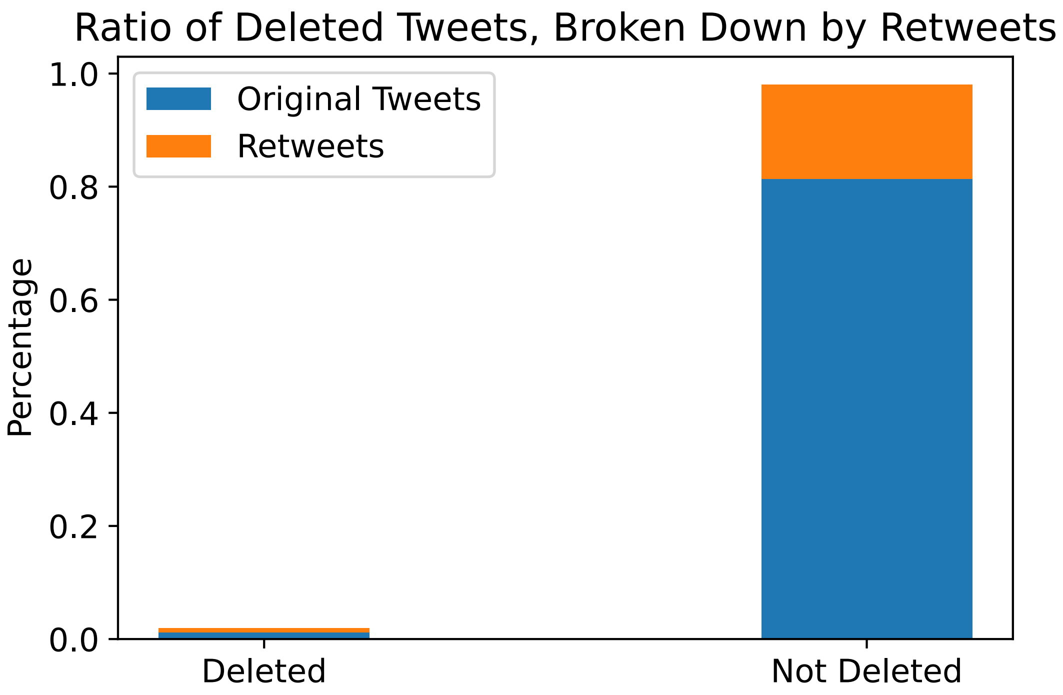
We can see that while the majority of tweets are from Trumps Offices' iPhone, Android, and desktop, there is a large variety of devices from which his tweets originate. This tells us that while the majority of the tweets likely came from his personal mobile devices over the years, there's a minority of tweets that likely came from an agent/communications director on his team.

Now let's see some of the ratio balances in our data. How many tweets were retweets, or deleted? Or how many were from certain devices?

In [9]:

```
labels = ['Deleted', 'Not Deleted']
df_deleted = data[data['isDeleted'] == 't']
df_notdeleted = data[data['isDeleted'] == 'f']
dfs = [df_deleted, df_notdeleted]
normal_tweet_ratios = []
retweet_ratios = []

for df in dfs:
    normal_tweet_ratio = df[df['isRetweet'] == 'f'].shape[0]/n
    retweet_ratio = df[df['isRetweet'] == 't'].shape[0]/n
    normal_tweet_ratios.append(normal_tweet_ratio)
    retweet_ratios.append(retweet_ratio)

#men_means = [20, 35, 30, 35, 27]
#women_means = [25, 32, 34, 20, 25]
width = 0.35       # the width of the bars: can also be len(x) sequence
fig, ax = plt.subplots()
print(str(retweet_ratios) + "\n" + str(normal_tweet_ratios))

ax.bar(labels, normal_tweet_ratios, width, label='Original Tweets')
ax.bar(labels, retweet_ratios, width, bottom=normal_tweet_ratios, label='Retweets')

ax.set_ylabel('Percentage')
ax.set_title('Ratio of Deleted Tweets, Broken Down by Retweets')
ax.legend()

plt.show()
fig.savefig("../figs/deletion_ratio.jpg")
```

```
[0.007760159799190398, 0.16683459723179722]
[0.011543016740025808, 0.8138622262289866]
```



We can see that the vast majority of Trump's tweets were not deleted, so we have an unbalanced dataset on our hands. In addition, among the deleted tweets, the balance between original tweets and retweets is close, while the balance of original tweets to retweets is majority original among the non-deleted tweets.

Now, let's take the topics from our LDA model and see if we can piece-together some patterns/themes

In [11]:

```
# tweets with topic labels
topic_modeled = pd.read_csv('../output/tweets_with_topic_label.csv')

# sentiment data from Olha's branch
sentiment_analysis_clean = pd.read_csv('../output/sentiment_analysis_clean.csv')
sentiment_labels = pd.read_csv('../output/sentiment_labels.csv')

# most frequent words in topics
topic_tops = pd.read_csv('../output/topic top words v2.csv')
```

In [12]:

```
topic_modeled['idx'] = topic_modeled.index
sentiment_labels.rename(columns={'Unnamed: 0':'idx'}, inplace=True)
```

In [13]:

```
topic_modeled = pd.merge(sentiment_labels,topic_modeled.drop(['id','text'], axis = 1),on='idx')
```

In [14]:

```
# Check if they are retweets: there are retweets in the middle of the text
tweets = topic_modeled['text'].to_list()
values = []

for tweet in tweets:
    if tweet.find('RT @') == -1:
        value = False
    else:
        value = True
    values.append(value)

topic_modeled['RT'] = values
```

In [15]:

```
print(len(sentiment_analysis_clean))
print(len(topic_modeled))
print(len(topic_modeled[topic_modeled.RT==False])) #if contains "RT @"
print(len(sentiment_analysis_clean[sentiment_analysis_clean.isRetweet=='f']))
print(len(sentiment_analysis_clean[sentiment_analysis_clean.retweeted==False])) #if startswith "RT"
```

```
54674
54422
44734
45133
45317
```

In [16]:

```
# Extract time from date
topic_modeled['date'] = topic_modeled.date.astype('datetime64[ns]')
#topic_modeled['year'] = topic_modeled.date.dt.year
#topic_modeled['day'] = topic_modeled.date.dt.date
#topic_modeled['month'] = topic_modeled.date.dt.month
topic_modeled['time'] = topic_modeled.date.dt.time
topic_modeled['hour'] = pd.to_numeric(topic_modeled.date.dt.hour)
```

In [17]:

```
[print(str(i),'\n',Counter(topic_modeled[i]),'\n','-'*30) for i in ['device','isDeleted','RT','Topic','hour','Final']
```

```
device
```

```
 Counter({'Twitter for iPhone': 25988, 'Twitter for Android': 14471, 'Twitter Web Client': 12122, 'TweetDeck': 481,
'TwitLonger Beta': 402, 'Twitter Media Studio': 347, 'Instagram': 133, 'Facebook': 105, 'Twitter Ads': 96, 'Twitter f
or BlackBerry': 96, 'Twitter Web App': 64, 'Twitter for iPad': 59, 'Twitlonger': 23, 'Twitter QandA': 10, 'Vine - Mak
e a Scene': 10, 'Periscope': 6, 'Neatly For BlackBerry 10': 5, 'Media Studio': 2, 'Twitter for Websites': 1, 'Twitter
Mirror for iPad': 1})
 ------------------------------
isDeleted
 Counter({'f': 53423, 't': 999})
 ------------------------------
RT
 Counter({False: 44734, True: 9688})
 ------------------------------
Topic
 Counter({0: 7519, 3: 7141, 9: 6895, 4: 5834, 6: 5337, 2: 4767, 1: 4745, 5: 4659, 7: 3809, 8: 3716})
 ------------------------------
hour
 Counter({12: 3656, 20: 3607, 19: 3408, 13: 3282, 11: 3027, 14: 2996, 18: 2930, 2: 2893, 15: 2805, 21: 2804, 1: 2649,
16: 2527, 3: 2518, 22: 2331, 0: 2270, 17: 2237, 23: 2133, 4: 1858, 10: 1571, 5: 956, 9: 646, 6: 528, 8: 431, 7: 359})
 ------------------------------
Final
 Counter({1: 33397, -1: 15045, 0: 5980})
 ------------------------------
```

Out[17]:

```
[None, None, None, None, None, None]
```

In [18]:

```
topic_modeled
```

Out[18]:

	idx	id	text	sentiment_text	subjectivity_score	VADER	TextBlob	W2V-kNN	Final	isRetweet	isDeleted
0	0	98454970654916608	Republicans and Democrats have both created ou...	republicans democrats created economic problems	0.200000	-1	1	-1	-1	f	f
1	1	1234653427789070336	I was thrilled to be back in the Great city of...	thrilled_back great city charlotte north_carol...	0.483333	1	1	-1	1	f	f
2	2	1218010753434820614	RT @CBS_Herridge: READ: Letter to surveillance...	read letter surveillance court obtained cbs ne...	0.100000	0	1	1	1	t	f

	idx	id	text	sentiment_text	subjectivity_score	VADER	TextBlob	W2V-kNN	Final	isRetweet	isDeleted
3	3	1304875170860015617	The Unsolicited Mail In Ballot Scam is a major...	unsolicited mail_ballot scam major threat demo...	0.454762	-1	1	-1	-1	f	f
4	4	1218159531554897920	RT @MZHemingway: Very friendly telling of even...	friendly telling events comey apparent leaking...	0.425000	1	1	-1	1	t	f
...	...	...	...	...	...	...	...	...	...	...	...
54417	54669	1319485303363571714	RT @RandPaul: I don't know why @JoeBiden think...	' know thinks continue lie wants ban_fracking ...	0.100000	-1	1	1	1	t	f
54418	54670	1319484210101379072	RT @EliseStefanik: President @realDonaldTrump ...	president excels communicating directly americ...	0.000000	1	0	1	1	t	f
54419	54671	1319444420861829121	RT @TeamTrump: LIVE: Presidential Debate #Deba...	live presidential_debate text vote	0.500000	0	1	-1	0	t	f
54420	54672	1319384118849949702	Just signed an order to support the workers of...	signed order support workers delphi corporatio...	0.260317	0	-1	-1	-1	t	f
54421	54673	1319345719829008387	Suburban women want Safety &amp; Security. Joe...	suburban women want safety_security joe_biden ...	0.000000	1	0	1	1	t	f

54422 rows × 20 columns

In [19]:

```
# save
topic_modeled.to_csv(r'../output/data_for_analysis.csv', index=False)
```

30 most frequent words in topics

In [20]:

```
# frequency ascending
for i in range(len(topic_tops)):
    print(topic_tops.iloc[i,1])
    print('-'*50)
```

```
[['wait', 'teamtrump', 'service', 'building', 'press', 'million', 'war', 'energy', 'truly', 'dont', 'lot', 'candidat
e', 'remember', 'open', 'presidential', 'end', 'problem', 'live', 'place', 'soon', 'doesnt', 'white', 'nation', 'whit
ehouse', 'sta', 'people', 'better', 'watch', 'look', 'house']]
--------------------------------------------------
[['absolutely', 'highest', 'given', 'robe', 'mark', 'miss', 'save', 'celebrity', 'win', 'russian', 'apprenticenbc',
'god', 'wow', 'ivankatrump', 'happen', 'stand', 'celebapprentice', 'apprentice', 'justice', 'case', 'senator', 'cou',
'witch', 'hunt', 'rating', 'book', 'congratulation', 'tonight', 'best', 'great']]
--------------------------------------------------
[['failing', 'success', 'texas', 'going', 'price', 'south', 'truth', 'cruz', 'tariff', 'drug', 'korea', 'federal', 'c
ompany', 'lie', 'schiff', 'lost', 'happy', 'york', 'hit', 'fantastic', 'iran', 'time', 'course', 'wall', 'border', 'f
bi', 'security', 'story', 'record', 'national']]
--------------------------------------------------
[['election', 'history', 'administration', 'didnt', 'far', 'dont', 'thats', 'best', 'usa', 'republican', 'russia', 'b
ad', 'right', 'night', 'said', 'impeachment', 'say', 'working', 'real', 'hard', 'democrat', 'foxnews', 'united', 'wor
ld', 'win', 'really', 'state', 'country', 'people', 'job']]
--------------------------------------------------
[['tower', 'criminal', 'sign', 'allowed', 'stock', 'comey', 'ive', 'voting', 'future', 'hotel', 'ready', 'market', 'p
otus', 'rally', 'government', 'golf', 'florida', 'john', 'makeamericagreatagain', 'billion', 'tomorrow', 'forward',
'man', 'people', 'got', 'work', 'make', 'donald', 'america', 'great']]
--------------------------------------------------
[['fraud', 'force', 'southern', 'phony', 'country', 'stay', 'ing', 'jim', 'disaster', 'voter', 'iowa', 'major', 'fo
x', 'dems', 'point', 'number', 'office', 'honor', 'interview', 'stop', 'election', 'democrat', 'let', 'poll', 'republ
ican', 'vote', 'repo', 'medium', 'fake', 'news']]
--------------------------------------------------
[['year', 'guy', 'carolina', 'city', 'million', 'political', 'ant', 'order', 'long', 'making', 'illegal', 'wonderfu
l', 'governor', 'woman', 'state', 'coming', 'crooked', 'maga', 'impo', 'life', 'looking', 'law', 'campaign', 'clinto
n', 'country', 'love', 'hillary', 'obama', 'great', 'american']]
--------------------------------------------------
[['depa', 'crazy', 'bad', 'donaldjtrumpjr', 'mean', 'believe', 'night', 'race', 'fighting', 'taking', 'entrepreneur',
'rate', 'seen', 'month', 'ago', 'tremendous', 'sad', 'friend', 'home', 'trying', 'word', 'change', 'join', 'mexico',
'debate', 'thing', 'cnn', 'going', 'true', 'year']]
--------------------------------------------------
[['politics', 'information', 'special', 'highly', 'approval', 'investigation', 'loser', 'used', 'interviewed', 'bus
h', 'youre', 'cont', 'told', 'joe', 'mike', 'general', 'corrupt', 'person', 'agree', 'team', 'trump', 'yesterday', 'm
ueller', 'enjoy', 'obamacare', 'morning', 'collusion', 'meeting', 'getting', 'foxandfriends']]
--------------------------------------------------
[['biden', 'dollar', 'sma', 'money', 'gop', 'leader', 'make', 'trade', 'week', 'crime', 'senate', 'america', 'amazin
g', 'congress', 'fact', 'family', 'economy', 'business', 'strong', 'military', 'total', 'vote', 'democrat', 'tax', 'b
order', 'suppo', 'china', 'deal', 'run', 'need']]
--------------------------------------------------
```

We can see some expected themes from the topic model. There is a topic regarding election fraud, with words such as "fraud, phony, vote, dems, fake and news". But, many topics share common themes. For example, Trumps MAGA slogen appears in several topics, as does "america", "democrat" and "vote/voter". This likely speaks to the overlap of prose in Trump's declarations; the topics are not cleanly segmeneted.

In [21]:

```
topic_labels = ['Whitehouse','Apprentice Show','National security','Election','MAGA',
                'Fake news','Hillary & Obama','President Trump','Interviews','China']
```

In [22]:

```
# 'realdonaldtrump' is occurs very frequently in topics
# so check if texts contain 'realdonaldtrump'
tweets = topic_modeled['text'].to_list()
values = []

for tweet in tweets:
    if tweet.find('realdonaldtrump') == -1:
        value = False
    else:
        value = True
    values.append(value)

topic_modeled['realdonaldtrump']= values
```

In [23]:

```
print(len(topic_modeled[(topic_modeled.realdonaldtrump==True)]))
print(len(topic_modeled[(topic_modeled.realdonaldtrump==True) & (topic_modeled.RT==False)].text))
```

```
123
81
```

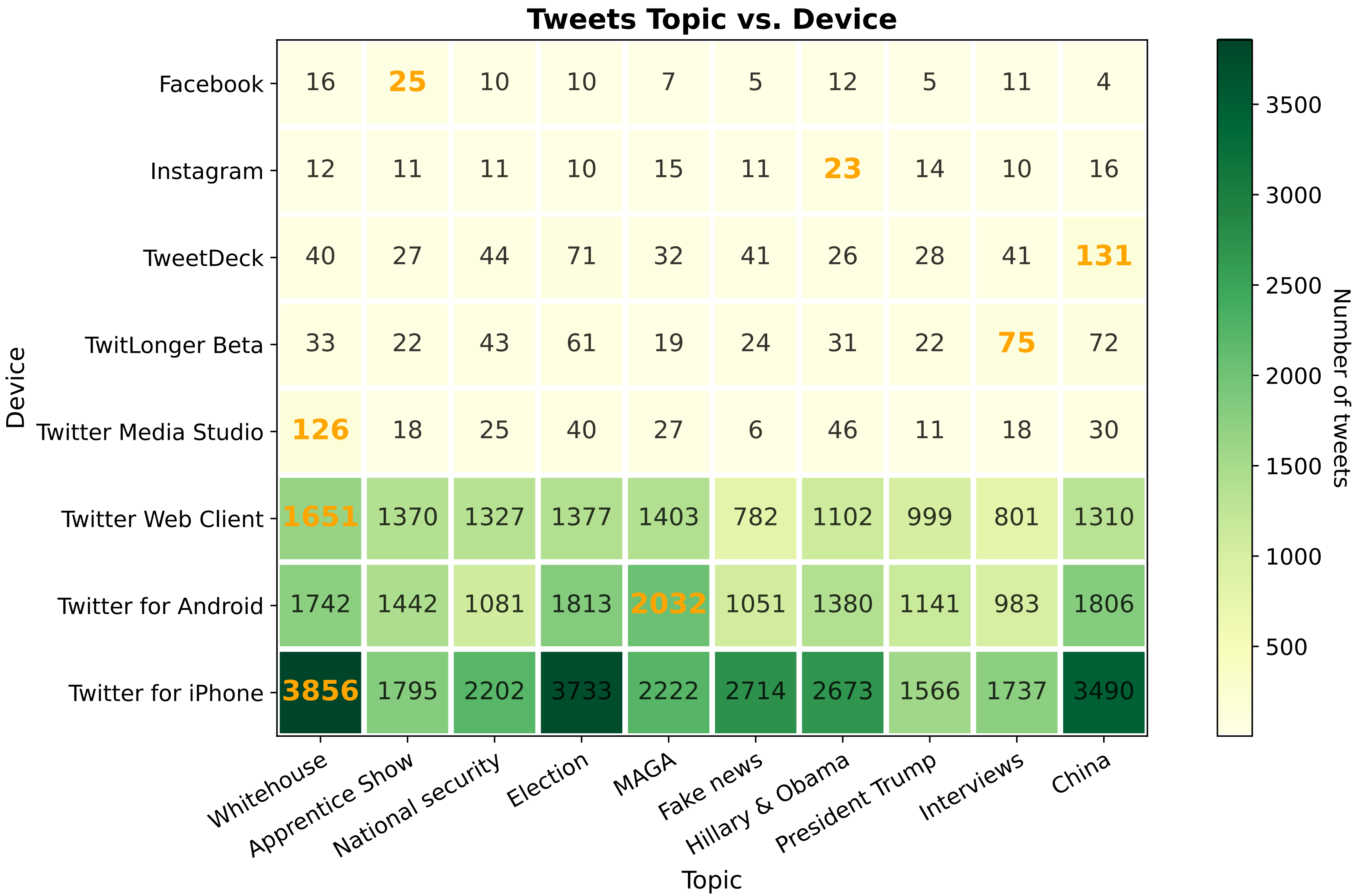
Heatmap: Tweet topic vs. device

In [24]:

```
# Reference: https://matplotlib.org/stable/gallery/images_contours_and_fields/image_annotated_heatmap.html
def heatmap(data, row_labels, col_labels, ax=None,
            cbar_kw={}, cbarlabel="", **kwargs):
    """
    Create a heatmap from a numpy array and two lists of labels.

    inputs
    ----------
    data
        A 2D numpy array of shape (N, M).
    row_labels
        A list or array of length N with the labels for the rows.
    col_labels
        A list or array of length M with the labels for the columns.
    ax
        A `matplotlib.axes.Axes` instance to which the heatmap is plotted.  If
        not provided, use current axes or create a new one.  Optional.
    cbar_kw
        A dictionary with arguments to `matplotlib.Figure.colorbar`.  Optional.
    cbarlabel
        The label for the colorbar.  Optional.

    """
    if not ax:
        ax = plt.gca()

    # Plot the heatmap
    im = ax.imshow(data, **kwargs)

    # Create colorbar
    cbar = ax.figure.colorbar(im, ax=ax, **cbar_kw)
    cbar.ax.set_ylabel(cbarlabel, rotation=-90, va="bottom")

    # We want to show all ticks...
    ax.set_xticks(np.arange(data.shape[1]))
    ax.set_yticks(np.arange(data.shape[0]))
    # ... and label them with the respective list entries.
    ax.set_xticklabels(col_labels)
    ax.set_yticklabels(row_labels)

    # Rotate the tick labels and set their alignment.
    plt.setp(ax.get_xticklabels(), rotation=30, ha="right",
             rotation_mode="anchor")

    ax.set_xticks(np.arange(data.shape[1]+1)-.5, minor=True)
    ax.set_yticks(np.arange(data.shape[0]+1)-.5, minor=True)
    ax.grid(which="minor", color="w", linestyle='-', linewidth=3)
    ax.tick_params(which="minor", bottom=False, left=False)

    for i in range(len(row_labels)):
        for j in range(len(col_labels)):
            if a[i,j] == np.max(a, axis=1)[i]:
                text = ax.text(j, i, a[i, j],fontsize=15,
                           ha="center", va="center", color="orange", weight="bold")
            else:
                text = ax.text(j, i, a[i, j],fontsize=13,
                           ha="center", va="center", color="black",alpha=0.8)

    return im, cbar
```

In [25]:
```
# (target) topic variable
topic_val = list(sorted(Counter(topic_modeled.Topic).keys()))
```

In [26]:
```
# device variable
device_val= list(sorted(Counter(topic_modeled.device).keys()))

counts=[]
for i in device_val:
    counts.append([i, sum(Counter(topic_modeled[topic_modeled.device==i]["Topic"]).values())])
df = pd.DataFrame(counts, columns=['device','counts'])#.sort_values(by='counts',ascending=False)[:10]
# Filters devices with 100+ tweets
device_val=list(df[df.counts > 100].device)

# heatmap matrix

a = np.empty((0,len(topic_labels)),int)
for i in device_val:
    counter = Counter(topic_modeled[topic_modeled.device==i]["Topic"])
    row = np.array([dict(counter).get(key, 0) for key in topic_val]).reshape(-1,10)
    #row = np.array(list(dict(sorted(counter.items())).values())).reshape(-1,10)
    a= np.append(a, row, axis=0)

fig, ax = plt.subplots(figsize=(12,7))

im, cbar = heatmap(a, device_val, topic_labels, ax=ax,
                   cmap="YlGn", cbarlabel="Number of tweets")

ax.set_title("Tweets Topic vs. Device", fontsize=15, fontweight='bold')
ax.set_xlabel("Topic", fontsize=13)
ax.set_ylabel("Device", fontsize=13)
fig.tight_layout()
plt.show()
```



	Whitehouse	Apprentice Show	National security	Election	MAGA	Fake news	Hillary & Obama	President Trump	Interviews	China
Facebook	16	25	10	10	7	5	12	5	11	4
Instagram	12	11	11	10	15	11	23	14	10	16
TweetDeck	40	27	44	71	32	41	26	28	41	131
TwitLonger Beta	33	22	43	61	19	24	31	22	75	72
Twitter Media Studio	126	18	25	40	27	6	46	11	18	30
Twitter Web Client	1651	1370	1327	1377	1403	782	1102	999	801	1310
Twitter for Android	1742	1442	1081	1813	2032	1051	1380	1141	983	1806
Twitter for iPhone	3856	1795	2202	3733	2222	2714	2673	1566	1737	3490

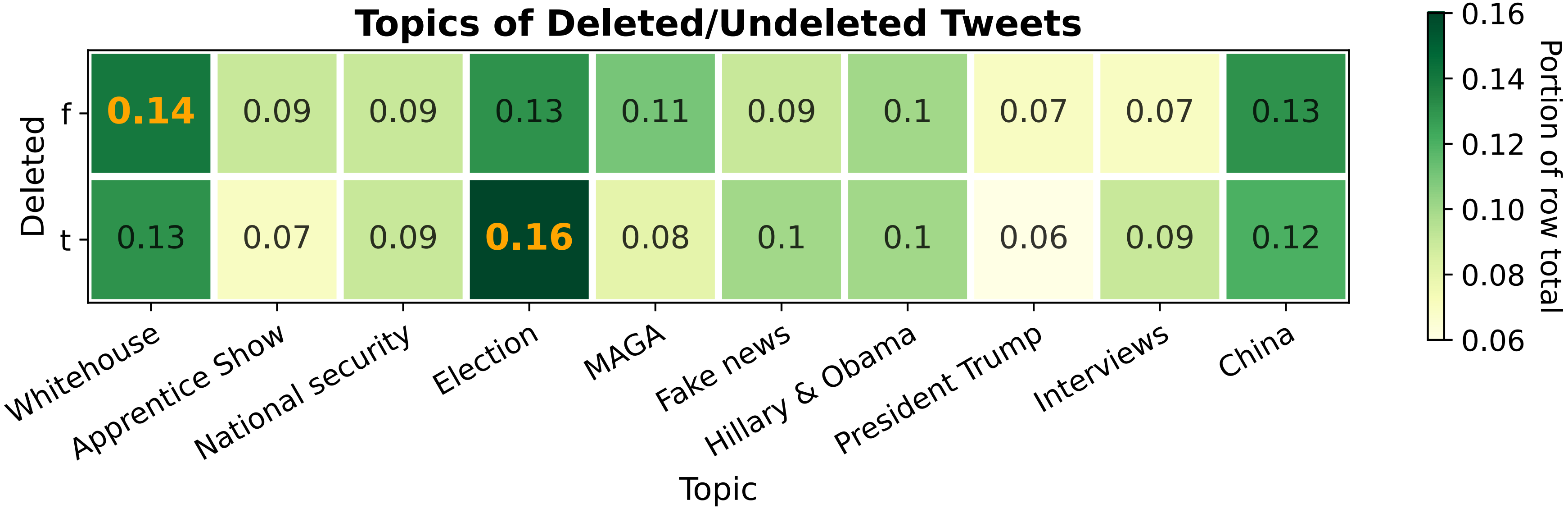
There's quite a lot going on in this heatmap, mainly because as we look across devices, we're not only looking at difference in topics by device, but also through time, since different devices were used during different time periods. So, for example, we can see that MAGA was a big topic on

Trump's android device in comparison to others, so he may have used an Android during his campaign. Similarly, whitehouse is a big topic from twitter media studio, so we can guess that perhaps that was someone on the whitehouse staff tweeting for him, or for him on the campaign trail.

Heatmap: Topics of Deleted/Undeleted Tweets

In [27]:

```
isdelete_val= list(sorted(Counter(topic_modeled.isDeleted).keys()))
# heatmap matrix

a = np.empty((0,len(topic_val)),int)
for i in isdelete_val:
    counter = Counter(topic_modeled[topic_modeled.isDeleted==i]["Topic"])
    row = np.array([dict(counter).get(key, 0) for key in topic_val]).reshape(-1,10)
    row = np.round(row/np.sum(row), decimals=2)
    #row = np.array(list(dict(sorted(counter.items())).values())).reshape(-1,10)
    a= np.append(a, row, axis=0)

fig, ax = plt.subplots(figsize=(10,3))

im, cbar = heatmap(a, isdelete_val, topic_labels, ax=ax,
                   cmap="YlGn", cbarlabel="Portion of row total")

ax.set_title("Topics of Deleted/Undeleted Tweets", fontsize=15, fontweight='bold')
ax.set_xlabel("Topic", fontsize=13)
ax.set_ylabel("Deleted", fontsize=13)
fig.tight_layout()
plt.show()
```



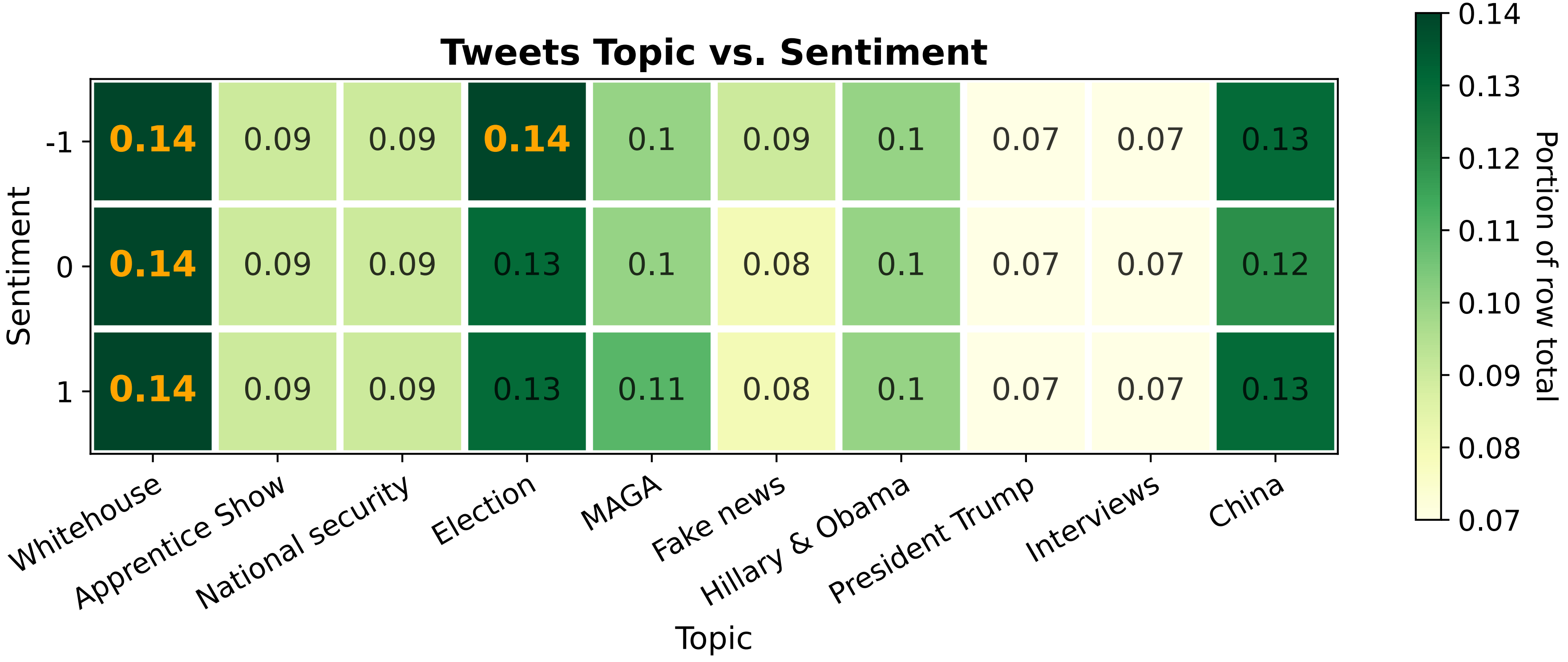
We immediatlely pick out what is expected in this graph: tweets relating to elections, and thereby Trump's claims of election fraud, were much more likely to be deleted due to Twitter's updated misinformation policies.

Heatmap: Tweets Topic vs. Sentiment

In [28]:

```
final_val= list(sorted(Counter(topic_modeled.Final).keys()))
# heatmap matrix

a = np.empty((0,len(topic_val)),int)
for i in final_val:
    counter = Counter(topic_modeled[topic_modeled.Final==i]["Topic"])
    row = np.array([dict(counter).get(key, 0) for key in topic_val]).reshape(-1,10)
    row = np.round(row/np.sum(row), decimals=2)
    #row = np.array(list(dict(sorted(counter.items())).values())).reshape(-1,10)
    a= np.append(a, row, axis=0)

fig, ax = plt.subplots(figsize=(10,4))

im, cbar = heatmap(a, final_val, topic_labels, ax=ax,
                   cmap="YlGn", cbarlabel="Portion of row total")
```

```
ax.set_title("Tweets Topic vs. Sentiment", fontsize=15, fontweight='bold')
ax.set_xlabel("Topic", fontsize=13)
ax.set_ylabel("Sentiment", fontsize=13)
fig.tight_layout()
plt.show()
```

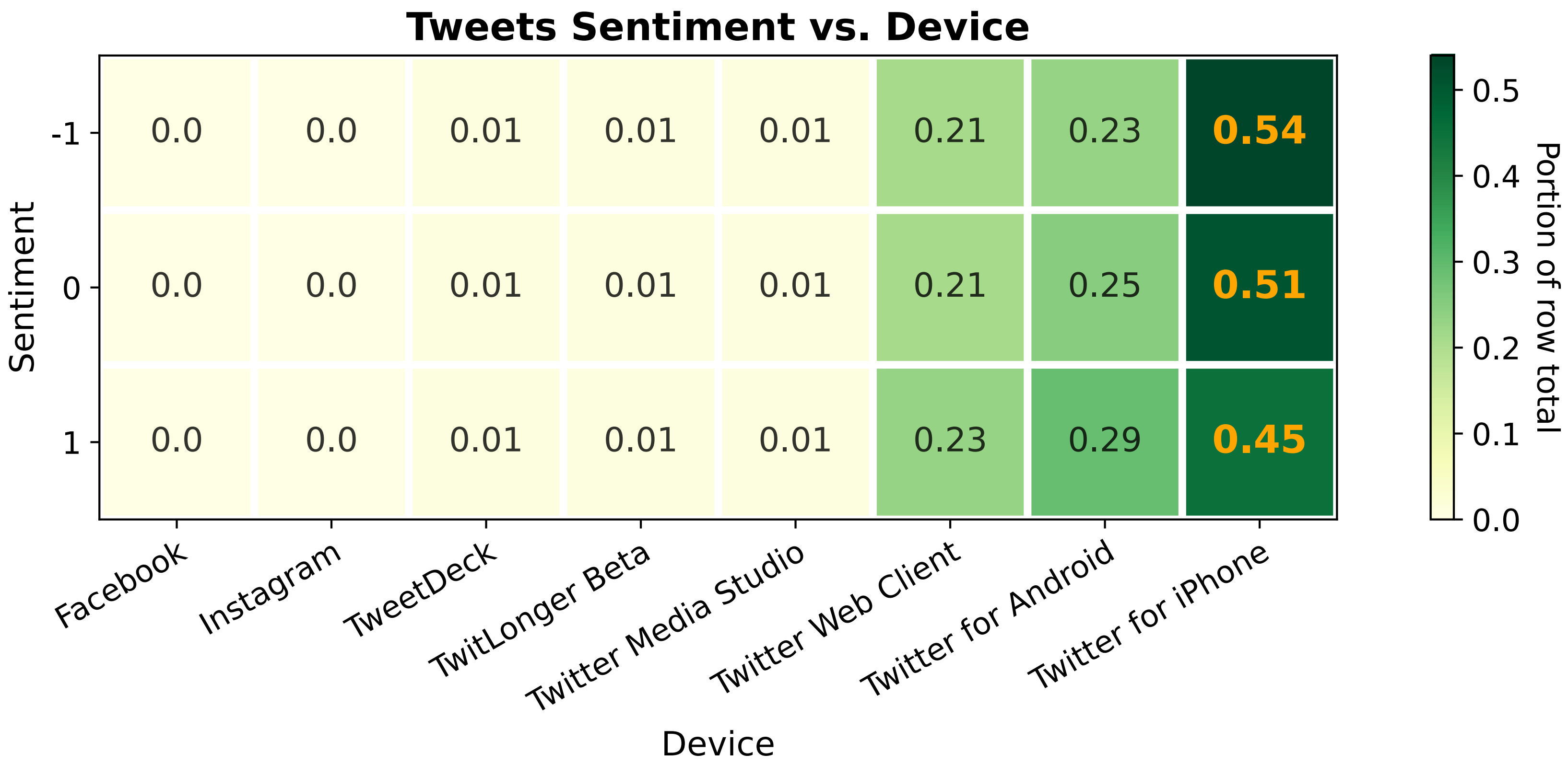


This is quite interesting. Most of these topics seem to have nearly equal sentiment value, meaning that these topics appear in positive/negative/neutral contexts all the same. Election topics are slightly more negative, but overall it seems that the sentiment is fairly balanced, whereas we may have expected political sentiment to bias towards negative.

Heatmap: Tweets Sentiment vs. Device

```
In [29]: final_val= list(sorted(Counter(topic_modeled.Final).keys()))
         # heatmap matrix
```

```
a = np.empty((0,len(device_val)),int)
for i in final_val:
    counter = Counter(topic_modeled[topic_modeled.Final==i]["device"])
    row = np.array([dict(counter).get(key, 0) for key in device_val]).reshape(-1,8)
    row = np.round(row/np.sum(row), decimals=2)
    #row = np.array(list(dict(sorted(counter.items())).values())).reshape(-1,10)
    a= np.append(a, row, axis=0)

fig, ax = plt.subplots(figsize=(11,4))

im, cbar = heatmap(a, final_val, device_val, ax=ax,
                   cmap="YlGn", cbarlabel="Portion of row total")


ax.set_title("Tweets Sentiment vs. Device", fontsize=15, fontweight='bold')
ax.set_xlabel("Device", fontsize=13)
ax.set_ylabel("Sentiment", fontsize=13)
fig.tight_layout()
plt.show()
```



Based on this heatmap we may assume that Trump used his iPhone for the majority of tweets during his campaign and presidency. Not only does that device have the majority of tweets overall, but also the sentiment biases negatively, and during his term his tweeting was infamous for its inflammatory nature.

WordCloud for each topic

In [30]:
```
delete = topic_modeled[topic_modeled.isDeleted=='t'].sentiment_text
len(delete)
```

Out[30]:
```
999
```

In [34]:
```
import cv2
path = "../output/"
```

```
# create image mask
img_grey = cv2.imread('../figs/trump.png', cv2.IMREAD_GRAYSCALE)
thresh = 240
# threshold the image
img_binary = cv2.threshold(img_grey, thresh, 255, cv2.THRESH_BINARY)[1]
#save image
cv2.imwrite(os.path.join(path, "trump_mask.png"),img_binary)
```

Out[34]: True

In [36]:

```
trump_mask = np.array(Image.open(os.path.join(path, "trump_mask.png")))
trump = np.array(Image.open('../figs/trump.png'))
image_colors = ImageColorGenerator(trump)
```

In [37]:

```
stopwords = set(STOPWORDS)
overused = ['thank','thanks','new','big','nice','like','time','year','years','know','think','thought',
            'want','good','little','never','wants','want','thing','follow','followed','go','going','way','love',
            'see','saw','high','low','say','says','day','today','different','realdonaldtrump','amp','true','really']
for i in overused:
    stopwords.add(i)
```

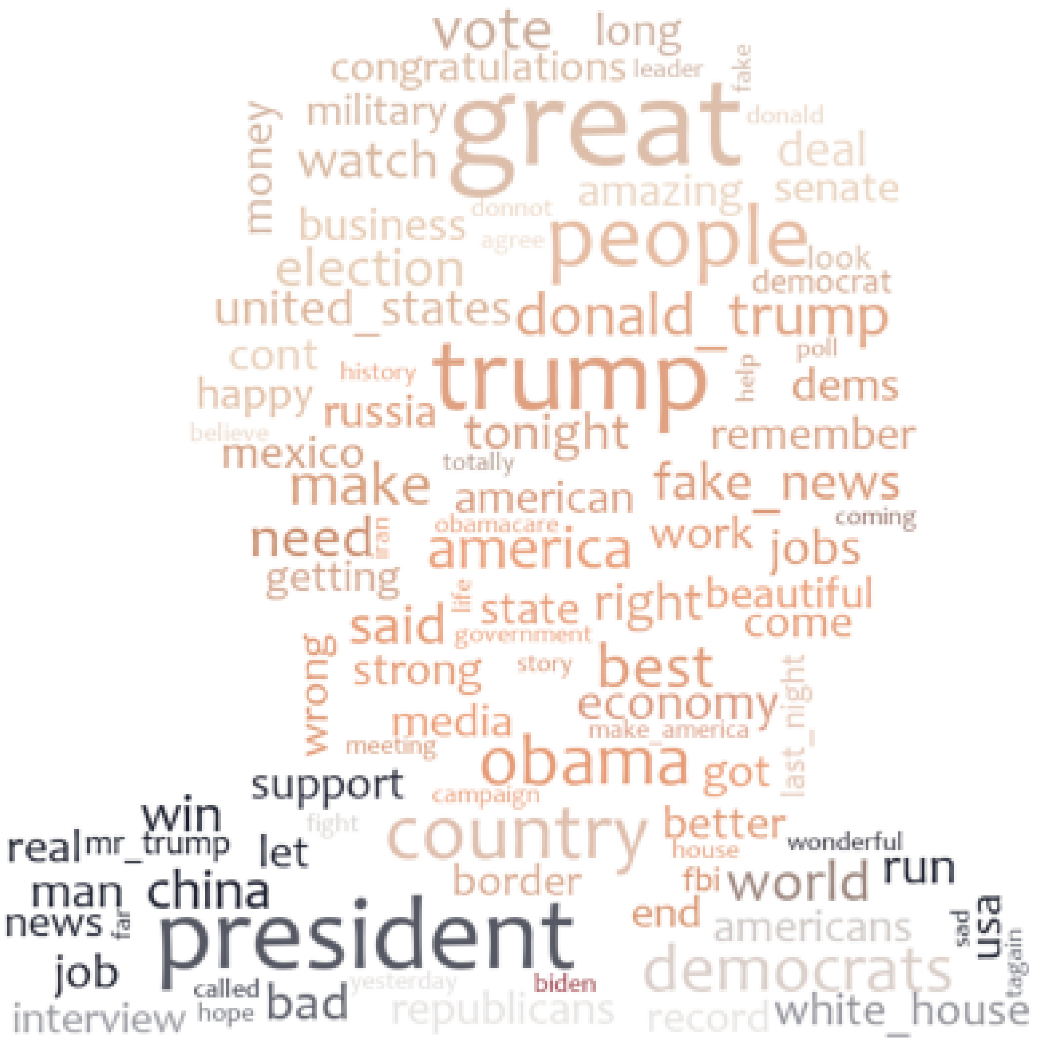
In [38]:

```
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer

def get_top_n_words(corpus, n=None):
    vec = CountVectorizer(stop_words='english').fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.items()]
    words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)
    return words_freq[:n]

def plot_wordcloud(topic=0, topic_label=''):
    freq_dict = dict(get_top_n_words(topic_modeled[topic_modeled.Topic==topic].sentiment_text))
    for i in stopwords:
        if i in freq_dict:
            freq_dict.pop(i)
    wordcloud = WordCloud(font_path='../data/Candara.ttf',
                          background_color='white',
                          max_words=100,
                          max_font_size=100,
                        mask=trump_mask,
                        random_state=42).generate_from_frequencies(freq_dict)
    fig = plt.figure(figsize=(8,8))
    #plt.imshow(wordcloud)
    plt.imshow(wordcloud.recolor(color_func=image_colors))#, interpolation="bilinear")
    plt.axis('off')
    plt.title('Topic {}: {}'.format(topic,topic_label), fontsize=15, fontweight='bold')
    plt.show()
```

In [39]:

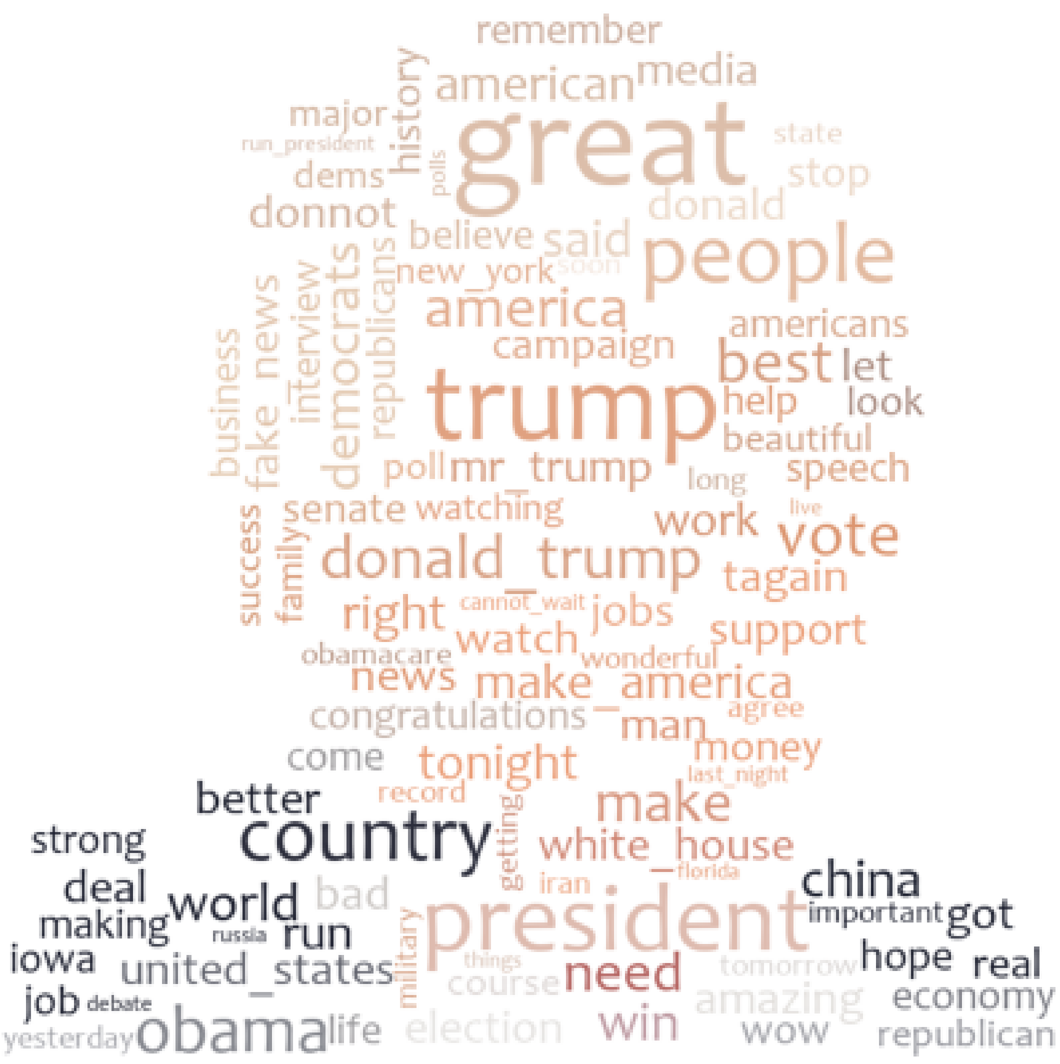
```
plot_wordcloud(topic=2, topic_label=topic_labels[2])
```

Topic 2: National security



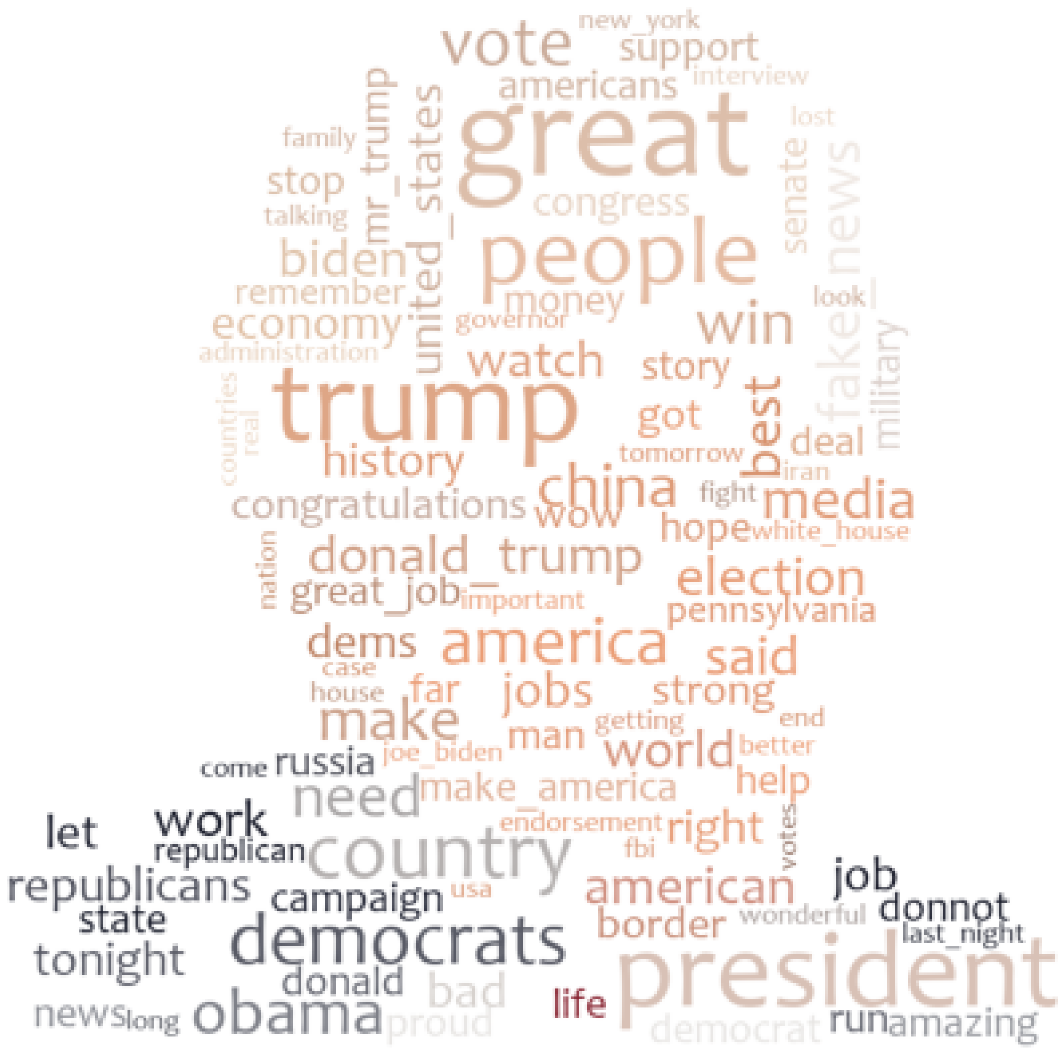
```
In [40]: plot_wordcloud(topic=4, topic_label=topic_labels[4])
```

Topic 4: MAGA



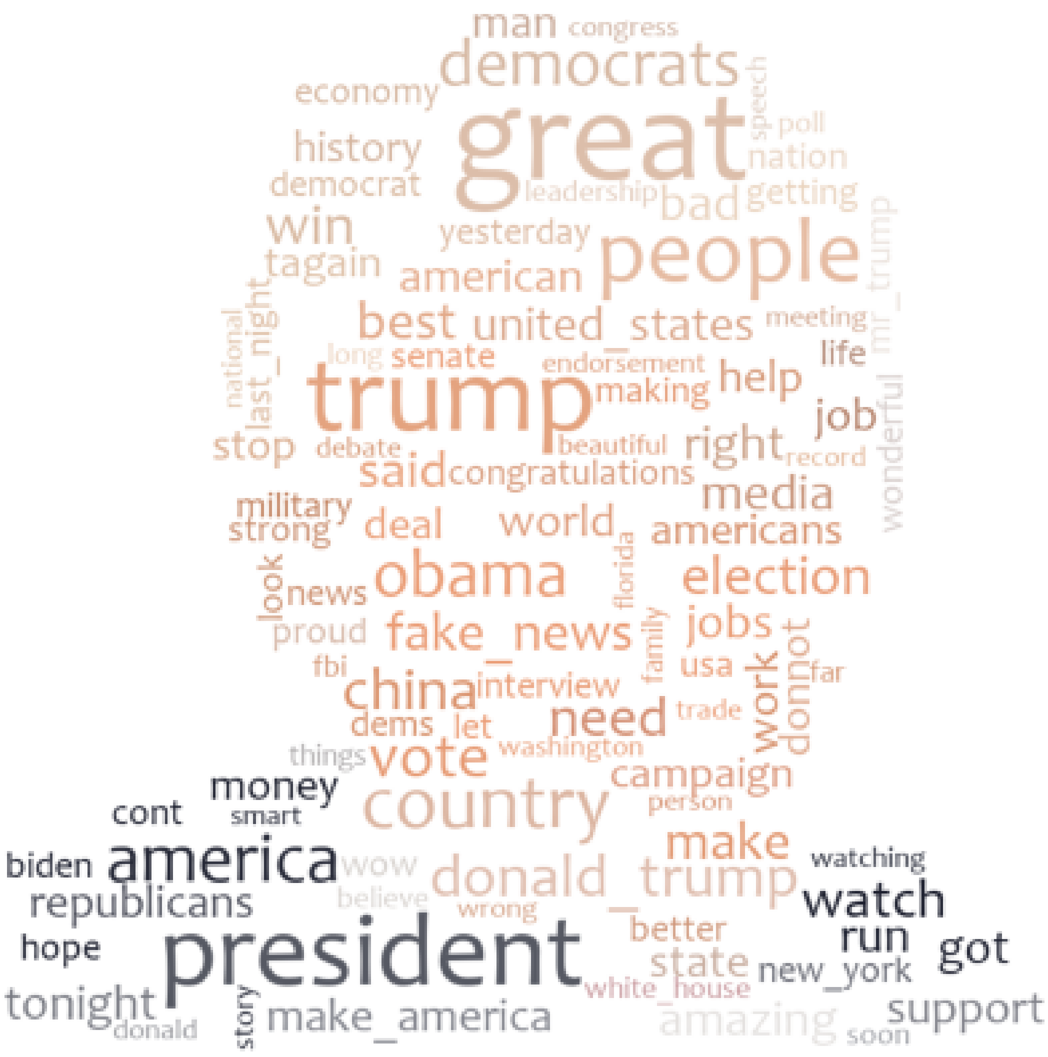
```
In [41]: plot_wordcloud(topic=5, topic_label=topic_labels[5])
```

Topic 5: Fake news



```
In [42]:  plot_wordcloud(topic=6, topic_label=topic_labels[6])
```

Topic 6: Hillary & Obama



In [ ]:

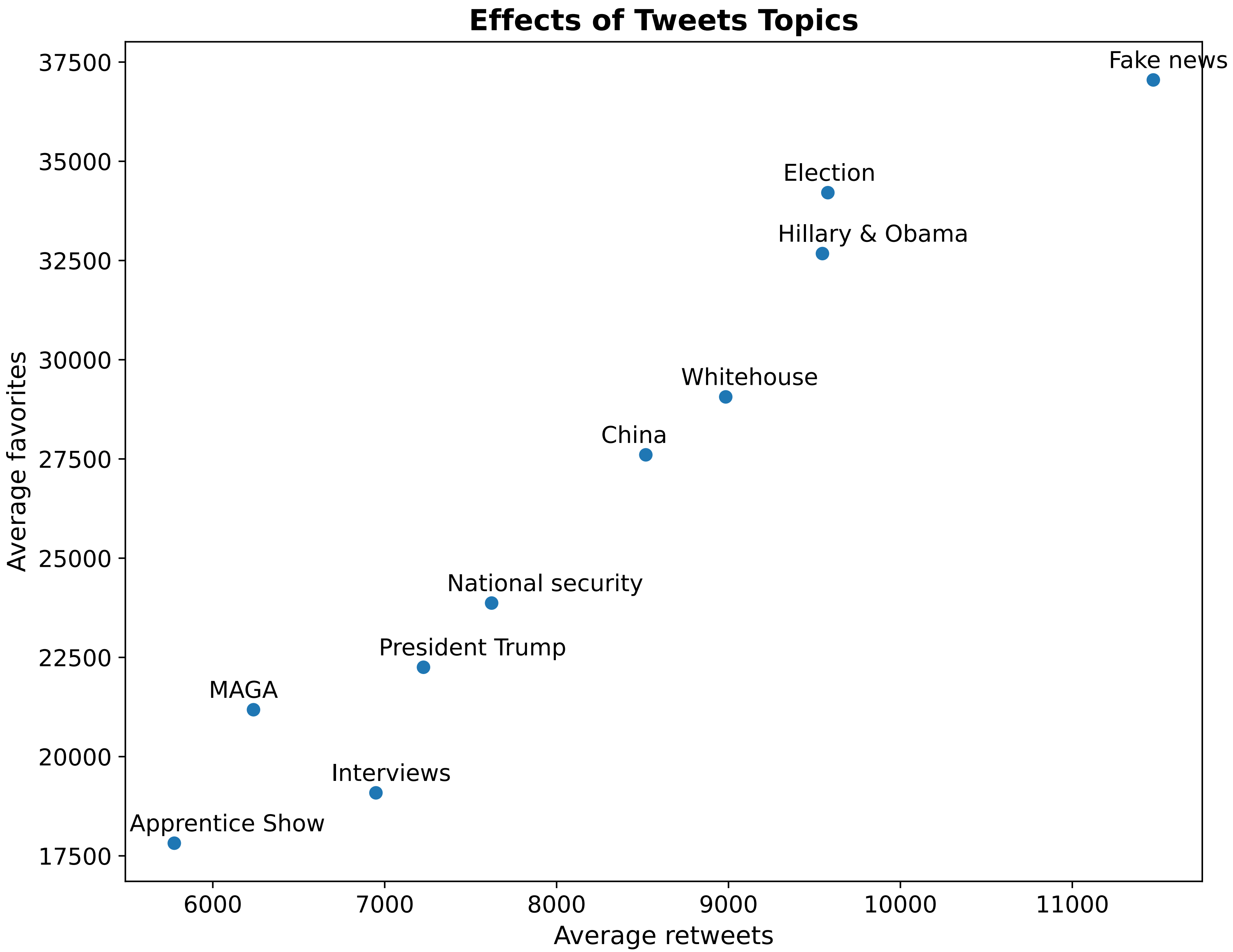
Effects of topics

In [43]:

```
avg_retweets = topic_modeled.groupby('Topic')['retweets'].agg(np.mean)
avg_favorites = topic_modeled.groupby('Topic')['favorites'].agg(np.mean)
#ratio = avg_retweets/avg_favorites

#df = np.array([topic_labels, list(avg_retweets), list(avg_favorites), list(ratio)])
#topic_summary = pd.DataFrame(df.T, columns=['topic','avg_retweets','avg_favorites','retweets_favorites_ratio'])
```

In [44]:

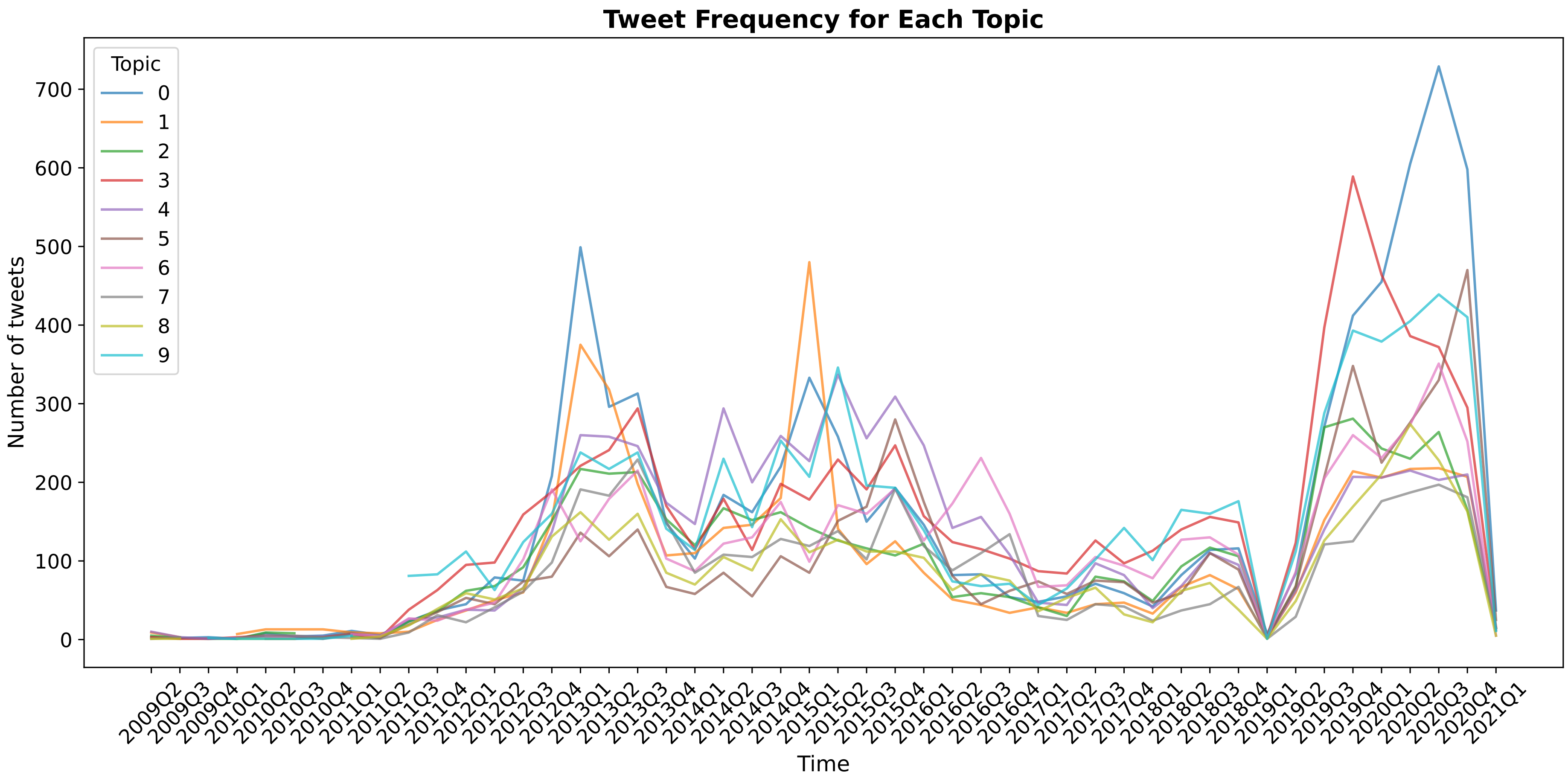
```
plt.figure(figsize=(10,8))
plt.scatter(avg_retweets, avg_favorites)
for i, txt in enumerate(topic_labels):
    plt.annotate(txt, (avg_retweets[i]-260, avg_favorites[i]+300))
plt.xlabel('Average retweets', fontsize=13)
plt.ylabel('Average favorites', fontsize=13)
plt.title('Effects of Tweets Topics', fontsize=15, fontweight='bold')
plt.show()
```


Effects of Tweets Topics
Fake news
Election
Hillary & Obama
Whitehouse
China
National security
President Trump
MAGA
Interviews
Apprentice Show
37500
35000
32500
30000
27500
25000
22500
20000
17500
Average favorites
6000
7000
8000
9000
10000
11000
Average retweets

Tweets about fake news have the highest average retweets and favorites; the Apprentice show tweets were published in earlier years before Trump was elected or campaigning, thus these tweets received the lowest social media engagement.

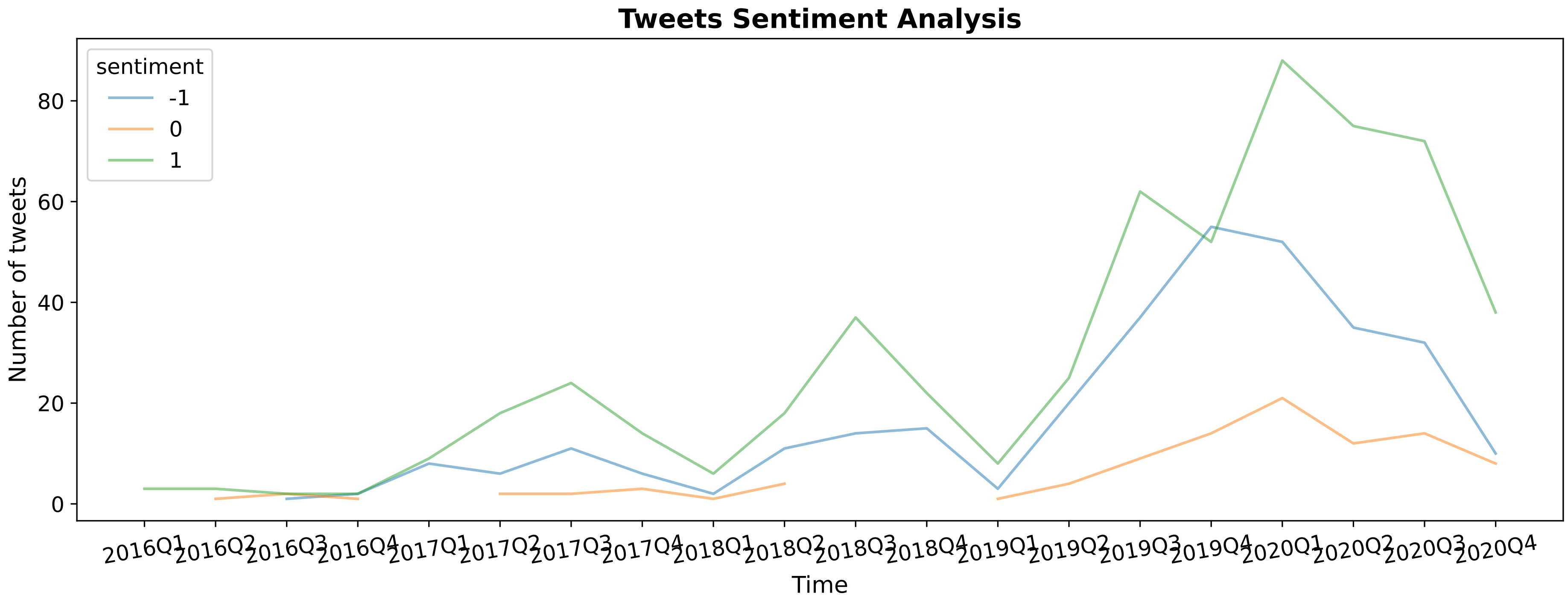
In [45]:

```
topic_modeled['yr-qt'] = topic_modeled.date.dt.year.astype(str) + 'Q' +topic_modeled.date.dt.quarter.astype(str)
topic_modeled['yr-m'] = topic_modeled.date.dt.year.astype(str) + '-' +topic_modeled.date.dt.month.astype(str)
# plot data
fig, ax = plt.subplots(figsize=(16,7))
# use unstack()
topic_modeled.groupby(['yr-qt','Topic'])['idx'].count().unstack().plot(ax=ax,alpha=0.7)#ls=('dashed'),
plt.xlabel('Time', fontsize=13)
plt.ylabel('Number of tweets', fontsize=13)
plt.title('Tweet Frequency for Each Topic', fontsize=15, fontweight='bold')
plt.xticks(np.arange(topic_modeled['yr-qt'].nunique()), np.sort(topic_modeled['yr-qt'].unique()), rotation=45)
plt.show()
```



In [46]:

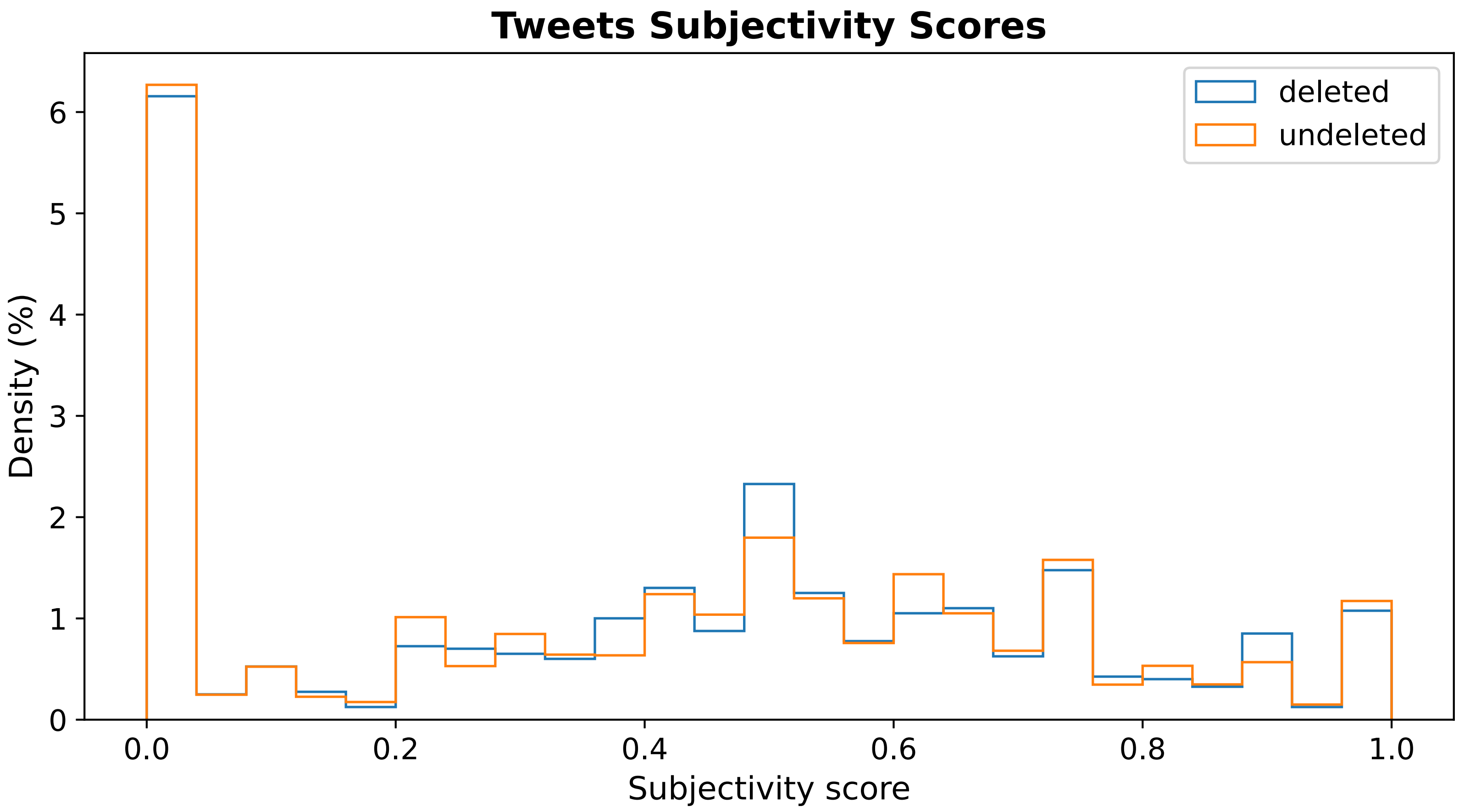
```python
# plot data
fig, ax = plt.subplots(figsize=(15,5))
# use unstack()

data = topic_modeled[topic_modeled.isDeleted=='t']
data['sentiment'] = data.Final
(data.groupby(['yr-qt','sentiment'])['idx'].count().unstack().plot(ax=ax,alpha=0.5))
plt.xlabel('Time', fontsize=13)
plt.ylabel('Number of tweets', fontsize=13)
plt.title('Tweets Sentiment Analysis', fontsize=15, fontweight='bold')
plt.xticks(np.arange(data['yr-qt'].nunique()), np.sort(data['yr-qt'].unique()), rotation=10)
plt.show()
```



While the number of tweets grew in 2020, the relative sentiment did not seem to change by much.

In [47]:

```
# No noticeable trend
deleted = topic_modeled[topic_modeled.isDeleted=='t']
undeleted = topic_modeled[topic_modeled.isDeleted=='f']
fig, ax = plt.subplots(figsize=(10,5))
plt.hist(deleted['subjectivity_score'], 25, histtype='step', stacked=True, fill=False,density = True, label='deleted
plt.hist(undeleted['subjectivity_score'], 25, histtype='step', stacked=True, fill=False,density = True, label='undele
plt.xlabel('Subjectivity score', fontsize=13)
plt.ylabel('Density (%)', fontsize=13)
plt.title('Tweets Subjectivity Scores', fontsize=15, fontweight='bold')
plt.legend()
plt.show()
```



Given how the distributions overlap, there actually doesn't appear to be a large difference between deleted and undeleted tweets by way of subjectivity, the only notable feature is that the deleted tweets have slightly more subjectivity scores ~0.5.

Imports and Reading Data

In [1]:

```
# libraries
import os, sys
import pandas as pd
import pickle
import re,string
import spacy,nltk
import sklearn,gensim,tweepy,pyLDAvis
from wordcloud import WordCloud
```

```
/Users/aprilyang/opt/anaconda3/lib/python3.8/site-packages/gensim/similarities/__init__.py:15: UserWarning: The gensi
m.similarities.levenshtein submodule is disabled, because the optional Levenshtein package <https://pypi.org/project/
python-Levenshtein/> is unavailable. Install Levenhstein (e.g. `pip install python-Levenshtein`) to suppress this war
ning.
  warnings.warn(msg)
/Users/aprilyang/.local/lib/python3.8/site-packages/sklearn/decomposition/_lda.py:28: DeprecationWarning: `np.float`
is a deprecated alias for the builtin `float`. To silence this warning, use `float` by itself. Doing this will not mo
dify any behavior and is safe. If you specifically wanted the numpy scalar type, use `np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecat
ions
  EPS = np.finfo(np.float).eps
```

In [2]:

```
# reference source: https://medium.datadriveninvestor.com/trump-tweets-topic-modeling-using-latent-dirichlet-allocati
df=pd.read_csv("../data/tweets_01-08-2021.csv")
df.head()
```

```
/Users/aprilyang/opt/anaconda3/lib/python3.8/site-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run
_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argu
ment and any exception that happen during thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

Out[2]:

	id	text	isRetweet	isDeleted	device	favorites	retweets	date	isFlagged
0	98454970654916608	Republicans and Democrats have both created ou...	f	f	TweetDeck	49	255	2011-08-02 18:07:48	f
1	1234653427789070336	I was thrilled to be back in the Great city of...	f	f	Twitter for iPhone	73748	17404	2020-03-03 01:34:50	f
2	1218010753434820614	RT @CBS_Herridge: READ: Letter to surveillance...	t	f	Twitter for iPhone	0	7396	2020-01-17 03:22:47	f

	id	text	isRetweet	isDeleted	device	favorites	retweets	date	isFlagged
3	1304875170860015617	The Unsolicited Mail In Ballot Scam is a major...	f	f	Twitter for iPhone	80527	23502	2020-09-12 20:10:58	f
4	1218159531554897920	RT @MZHemingway: Very friendly telling of even...	t	f	Twitter for iPhone	0	9081	2020-01-17 13:13:59	f

In [3]:

```
tweets_df=df.loc[:,['text']]
tweets_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 56571 entries, 0 to 56570
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   text    56571 non-null  object
dtypes: object(1)
memory usage: 442.1+ KB
/Users/aprilyang/opt/anaconda3/lib/python3.8/site-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run
_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argu
ment and any exception that happen during thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

In [4]:

```
# quick glance at the text
a = 56500
for i in range(a,a+10):
    print(tweets_df.text[i])
    print()
```

```
Great support coming from all sides for Border Security (including Wall) on our very dangerous Southern Border. Teams
negotiating this weekend! Washington Post and NBC reporting of events, including Fake sources, has been very inaccura
te (to put it mildly)!

Thank you to Kanye West for your nice words. Criminal Justice Reform is now law - passed in a very bipartisan way!

Great new book by Dr. Robert Jeffress, “Choosing the Extraordinary Life.” Get it and enjoy!  @LouDobbs

The story in the New York Times regarding Jim Webb being considered as the next Secretary of Defense is FAKE NEWS.
I’m sure he is a fine man, but I don’t know Jim, and never met him. Patrick Shanahan, who is Acting Secretary of Defe
nse, is doing a great job!

GREAT JOBS NUMBERS JUST ANNOUNCED!
```

```
How do you impeach a president who has won perhaps the greatest election of all time, done nothing wrong (no Collusio
n with Russia, it was the Dems that Colluded), had the most successful first two years of any president, and is the m
ost popular Republican in party history 93%?

As I have stated many times, if the Democrats take over the House or Senate, there will be disruption to the Financia
l Markets. We won the Senate, they won the House. Things will settle down. They only want to impeach me because they
know they can’t win in 2020, too much success!

....President Trump deserves a lot of credit, but again, you have the anti-Trump people who are not going to give him
a lot of credit.”

Michael Pillsbury interviewed by @cvpayne: “They have the motive of making the President look bad — instead of Presid
ent Trump being portrayed as a HERO. The first President to take China on, it’s 20 years overdue....

https://t.co/jsOrDtwdEa

/Users/aprilyang/opt/anaconda3/lib/python3.8/site-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run
_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argu
ment and any exception that happen during thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

Data Cleaing -- Using code from Mark

In [5]:

```
# Data cleaning:
# Lowercase text
# Remove brackets using regular expressions
# remove punctuation and numbers using regular expressions

def clean_text(text):
    text = text.lower()
    text = re.sub(r"http\S+", "", text)
    text = re.sub(r'\[.*?\]', '', text)

    text = re.sub(r'[^\w\s]', '', text)
    text = re.sub("https?://([^\s]+)", ' ', text) # links
    text = re.sub(r'^https?:\/\/.*[\r\n]*', '', text)

    text = re.sub("rt", ' ', text) # RT :
    text = re.sub(" &amp", ' ', text) # &amp
    text = re.sub("[\n\r\t\0]", ' ', text) # new line, tabs, etc
    text = re.sub('[!,.-;:\"“”\[\]{}]', ' ', text) # punct
    text = re.sub('\s{2,}', ' ', text) # 2+ whitespaces
    return text

tweets_df_clean = pd.DataFrame(tweets_df.text.apply(lambda x: clean_text(x)))
```

```
/Users/aprilyang/opt/anaconda3/lib/python3.8/site-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run
_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argu
ment and any exception that happen during thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
<>:12: DeprecationWarning: invalid escape sequence \s
<>:18: DeprecationWarning: invalid escape sequence \[
<>:19: DeprecationWarning: invalid escape sequence \s
<>:12: DeprecationWarning: invalid escape sequence \s
<>:18: DeprecationWarning: invalid escape sequence \[
<>:19: DeprecationWarning: invalid escape sequence \s
<ipython-input-5-17f1bc126e7b>:12: DeprecationWarning: invalid escape sequence \s
  text = re.sub("https?://([^\s]+)", ' ', text) # links
<ipython-input-5-17f1bc126e7b>:18: DeprecationWarning: invalid escape sequence \[
  text = re.sub('[!,.-;:\"“”\[\]{}]', ' ', text) # punct
<ipython-input-5-17f1bc126e7b>:19: DeprecationWarning: invalid escape sequence \s
  text = re.sub('\s{2,}', ' ', text) # 2+ whitespaces
```

In [6]:

```
tweets_df_clean.head()
```

```
/Users/aprilyang/opt/anaconda3/lib/python3.8/site-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run
_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argu
ment and any exception that happen during thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

Out[6]:

	text
0	republicans and democrats have both created ou...
1	i was thrilled to be back in the great city of...
2	cbs_herridge read letter to surveillance cou ...
3	the unsolicited mail in ballot scam is a major...
4	mzhemingway very friendly telling of events h...

In [7]:

```
from nltk.corpus import wordnet
import nltk
from nltk.corpus import stopwords
from nltk.stem.wordnet import WordNetLemmatizer
```

```
# nltk.download('wordnet') ##################### need this line or will get error -- this line takes forever to run
import string
```

```
/Users/aprilyang/opt/anaconda3/lib/python3.8/site-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run
_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argu
ment and any exception that happen during thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

In [8]:

```
# Perform Lemmatization to reduce inflected words to their root words.----cannot run, my computer failed :(
# nlp = spacy.load("en_core_web_sm")
# def lemmatizer(text):
#     sent = []
#     doc = nlp(text)
#     for word in doc:
#         sent.append(word.lemma_)
#     return " ".join(sent)

from nltk.corpus import wordnet

stop = set(stopwords.words('english'))
# manually adding stopwords
overused = ['thank','thanks','president','new','big','nice','like','time','year','know','think','thought',
            'want','good','little','never','wants','want','thing','follow','followed','go','way',
            'see','high','low','says','day','today','different','realdonaldtrump','amp','trump']
for i in overused:
    stop.add(i)


def lemmatizer(doc):
    lemma = WordNetLemmatizer()
    stop_free = " ".join([i for i in doc.lower().split() if i not in stop])
#     punc_free = ''.join(ch for ch in stop_free if ch not in exclude)
    normalized = " ".join(lemma.lemmatize(word) for word in stop_free.split())
    return normalized

tweets_df_clean = pd.DataFrame(tweets_df_clean.text.apply(lambda x: lemmatizer(x)))
tweets_df_clean['text'] = tweets_df_clean['text'].str.replace('-PRON-', '')
```

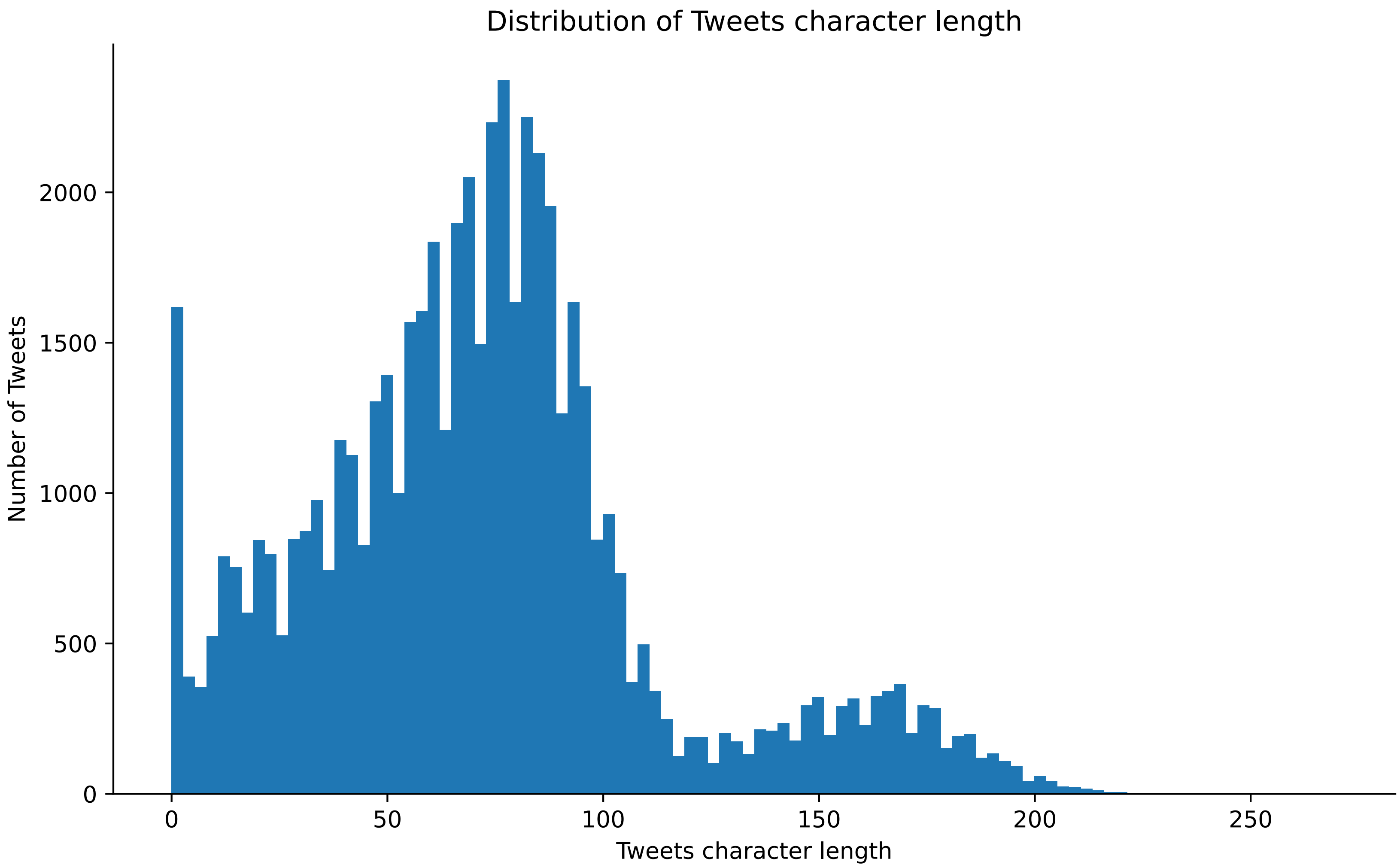
```
/Users/aprilyang/opt/anaconda3/lib/python3.8/site-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run
_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argu
ment and any exception that happen during thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

Easy EDAs

```
In [9]:
```

```python
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns

plt.figure(figsize=(10,6))
doc_lens = [len(d) for d in tweets_df_clean.text]
plt.hist(doc_lens, bins = 100)
plt.title('Distribution of Tweets character length')
plt.ylabel('Number of Tweets')
plt.xlabel('Tweets character length')
sns.despine();
```

```
/Users/aprilyang/opt/anaconda3/lib/python3.8/site-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run
_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argu
ment and any exception that happen during thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
/Users/aprilyang/opt/anaconda3/lib/python3.8/site-packages/ipykernel/pylab/config.py:70: DeprecationWarning: InlineBa
ckend._figure_formats_changed is deprecated in traitlets 4.1: use @observe and @unobserve instead.
  def _figure_formats_changed(self, name, old, new):
```

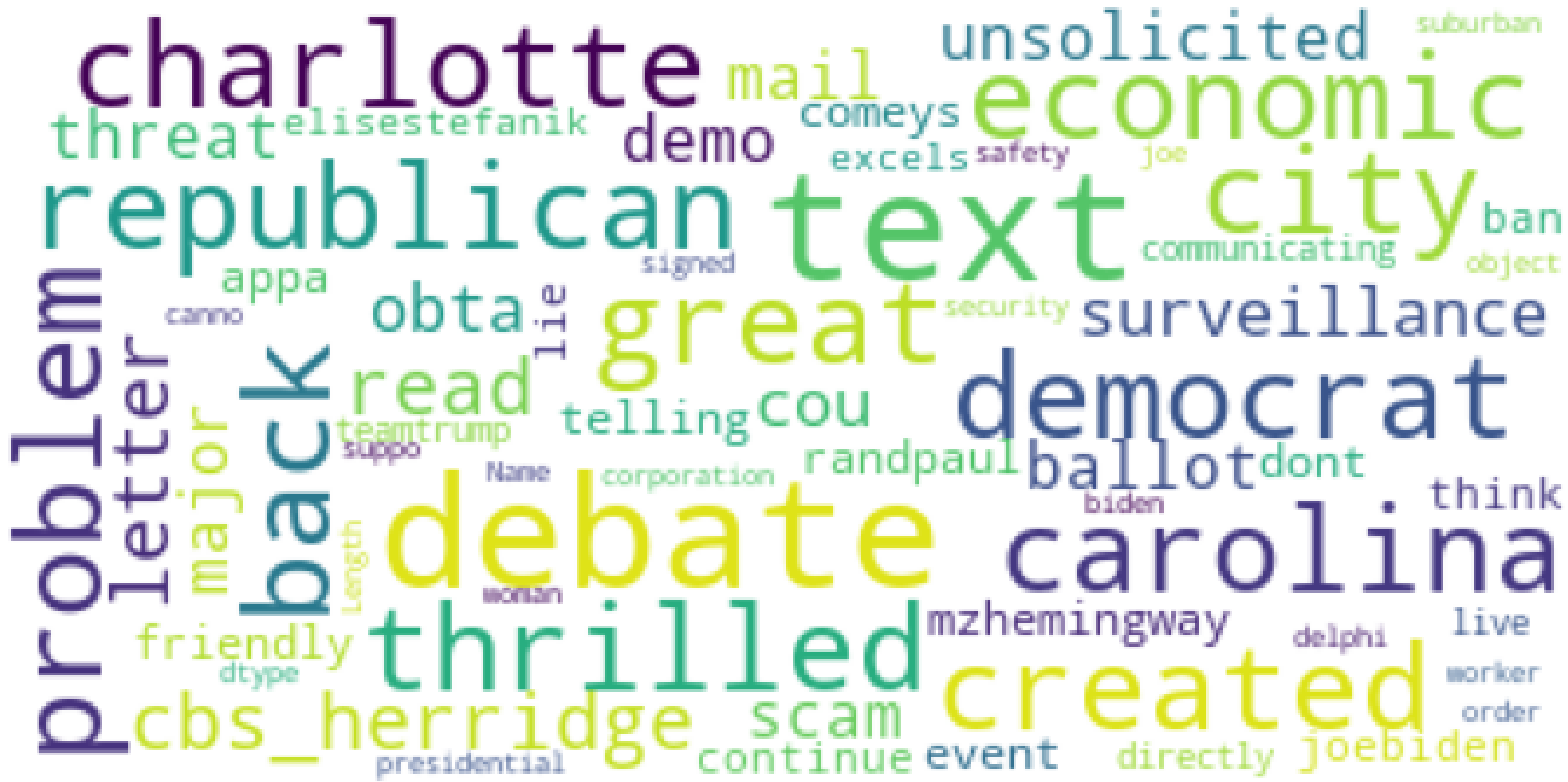


In [10]:

```
import matplotlib as mpl
from subprocess import check_output
from wordcloud import WordCloud, STOPWORDS

mpl.rcParams['figure.figsize']=(12.0,12.0)
mpl.rcParams['font.size']=12
mpl.rcParams['savefig.dpi']=100
mpl.rcParams['figure.subplot.bottom']=.1
stopwords = set(STOPWORDS)
```

```
wordcloud = WordCloud(
                          background_color='white',
                          stopwords=stopwords,
                          max_words=500,
                          max_font_size=40,
                          random_state=100
                         ).generate(str(tweets_df_clean.text))
print(wordcloud)
fig = plt.figure(1)
plt.imshow(wordcloud)
plt.axis('off')
plt.show();
```

```
/Users/aprilyang/opt/anaconda3/lib/python3.8/site-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run
_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argu
ment and any exception that happen during thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
<wordcloud.wordcloud.WordCloud object at 0x7f831cdb9f10>
```



Unigrams: remove all stop words to get unigrams

In [11]:

```
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer

def get_top_n_words(corpus, n=None):
    vec = CountVectorizer(stop_words='english').fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in vec.vocabulary_.items()]
    words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)
    return words_freq[:n]
```

```
common_words = get_top_n_words(tweets_df_clean.text, 10)
unigram = pd.DataFrame(common_words, columns = ['unigram' , 'count'])
unigram
```

```
/Users/aprilyang/opt/anaconda3/lib/python3.8/site-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run
_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argu
ment and any exception that happen during thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

Out[11]:

	unigram	count
0	great	7578
1	people	3507
2	country	2750
3	america	2441
4	democrat	2403
5	job	2324
6	state	2103
7	make	2061
8	news	2056
9	american	2050

Trigrams

In [12]:

```
def get_top_n_trigram(corpus, n=None):
    vec = CountVectorizer(ngram_range=(3,3),stop_words='english').fit(corpus)
    bag_of_words = vec.transform(corpus)
    sum_words = bag_of_words.sum(axis=0)
    words_freq = [(word, sum_words[0, idx]) for word, idx in      vec.vocabulary_.items()]
    words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)
    return words_freq[:n]

common_words = get_top_n_trigram(tweets_df_clean.text, 10)
trigram = pd.DataFrame(common_words, columns = ['trigram' , 'count'])
trigram
```

```
/Users/aprilyang/opt/anaconda3/lib/python3.8/site-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run
```

```
_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argu
ment and any exception that happen during thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

Out[12]:

	trigram	count
0	make america great	595
1	fake news medium	259
2	complete total endorsement	249
3	happy bi hday	149
4	crooked hillary clinton	137
5	radical left democrat	114
6	sleepy joe biden	89
7	let make america	84
8	strong crime border	72
9	fake news cnn	71

Topic modeling with LDA

In [13]:

```python
from sklearn.decomposition import LatentDirichletAllocation
vectorizer = CountVectorizer(
analyzer='word',
min_df=3,# minimum required occurences of a word
stop_words='english',# remove stop words
lowercase=True,# convert all words to lowercase
token_pattern='[a-zA-Z0-9]{3,}',# num chars > 3
max_features=5000,# max number of unique words
                            )

data_matrix = vectorizer.fit_transform(tweets_df_clean.text)
data_matrix
```

```
/Users/aprilyang/opt/anaconda3/lib/python3.8/site-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run
_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argu
ment and any exception that happen during thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

```
<56571x5000 sparse matrix of type '<class 'numpy.int64'>'
```

```
Out[13]:         with 422455 stored elements in Compressed Sparse Row format>
```

```
In [14]: lda_model = LatentDirichletAllocation(
         n_components=10, # Number of topics
         learning_method='online',
         random_state=20,
         n_jobs = -1  # Use all available CPUs
                                             )

         lda_output = lda_model.fit_transform(data_matrix)
```

```
/Users/aprilyang/opt/anaconda3/lib/python3.8/site-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run
_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argu
ment and any exception that happen during thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

```
In [15]: # !pip install pyLDAvis
         import pyLDAvis
         import pyLDAvis.sklearn
         pyLDAvis.enable_notebook()
         p=pyLDAvis.sklearn.prepare(lda_model, data_matrix, vectorizer, mds='tsne')
         p=pyLDAvis.save_html(p, '../output/lda_original_v2.html')
```

```
/Users/aprilyang/opt/anaconda3/lib/python3.8/site-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run
_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argu
ment and any exception that happen during thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
/Users/aprilyang/.local/lib/python3.8/site-packages/sklearn/metrics/pairwise.py:58: DeprecationWarning: `np.float` is
a deprecated alias for the builtin `float`. To silence this warning, use `float` by itself. Doing this will not modif
y any behavior and is safe. If you specifically wanted the numpy scalar type, use `np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecat
ions
  dtype = np.float
/Users/aprilyang/.local/lib/python3.8/site-packages/sklearn/manifold/_t_sne.py:349: DeprecationWarning: `np.float` is
a deprecated alias for the builtin `float`. To silence this warning, use `float` by itself. Doing this will not modif
y any behavior and is safe. If you specifically wanted the numpy scalar type, use `np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecat
ions
  error = np.finfo(np.float).max
/Users/aprilyang/.local/lib/python3.8/site-packages/sklearn/manifold/_t_sne.py:350: DeprecationWarning: `np.float` is
a deprecated alias for the builtin `float`. To silence this warning, use `float` by itself. Doing this will not modif
y any behavior and is safe. If you specifically wanted the numpy scalar type, use `np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecat
```

```
ions
  best_error = np.finfo(np.float).max
/Users/aprilyang/.local/lib/python3.8/site-packages/sklearn/manifold/_t_sne.py:349: DeprecationWarning: `np.float` is
a deprecated alias for the builtin `float`. To silence this warning, use `float` by itself. Doing this will not modif
y any behavior and is safe. If you specifically wanted the numpy scalar type, use `np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecat
ions
  error = np.finfo(np.float).max
/Users/aprilyang/.local/lib/python3.8/site-packages/sklearn/manifold/_t_sne.py:350: DeprecationWarning: `np.float` is
a deprecated alias for the builtin `float`. To silence this warning, use `float` by itself. Doing this will not modif
y any behavior and is safe. If you specifically wanted the numpy scalar type, use `np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecat
ions
  best_error = np.finfo(np.float).max
```

In [16]:

```
for i,topic in enumerate(lda_model.components_):
    print(f'Top 10 words for topic #{i}:')
    print([vectorizer.get_feature_names()[i] for i in topic.argsort()[-10:]])
    print('\n')
```

```
/Users/aprilyang/opt/anaconda3/lib/python3.8/site-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run
_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argu
ment and any exception that happen during thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
Top 10 words for topic #0:
['doesnt', 'white', 'nation', 'whitehouse', 'sta', 'people', 'better', 'watch', 'look', 'house']


Top 10 words for topic #1:
['senator', 'cou', 'witch', 'hunt', 'rating', 'book', 'congratulation', 'tonight', 'best', 'great']


Top 10 words for topic #2:
['iran', 'time', 'course', 'wall', 'border', 'fbi', 'security', 'story', 'record', 'national']


Top 10 words for topic #3:
['democrat', 'foxnews', 'united', 'world', 'win', 'really', 'state', 'country', 'people', 'job']


Top 10 words for topic #4:
['tomorrow', 'forward', 'man', 'people', 'got', 'work', 'make', 'donald', 'america', 'great']
```

```
Top 10 words for topic #5:
['election', 'democrat', 'let', 'poll', 'republican', 'vote', 'repo', 'medium', 'fake', 'news']


Top 10 words for topic #6:
['looking', 'law', 'campaign', 'clinton', 'country', 'love', 'hillary', 'obama', 'great', 'american']


Top 10 words for topic #7:
['word', 'change', 'join', 'mexico', 'debate', 'thing', 'cnn', 'going', 'true', 'year']


Top 10 words for topic #8:
['trump', 'yesterday', 'mueller', 'enjoy', 'obamacare', 'morning', 'collusion', 'meeting', 'getting', 'foxandfriend
s']


Top 10 words for topic #9:
['total', 'vote', 'democrat', 'tax', 'border', 'suppo', 'china', 'deal', 'run', 'need']
```

In [18]:

```python
# get top 30 words in each topic and generate a csv file
df_topic=pd.DataFrame(columns=['Topic','TopWords'])
word_list=[]

for i,topic in enumerate(lda_model.components_):
    sub_list=[]
    print(f'Top 30 words for topic #{i}:')
    print([vectorizer.get_feature_names()[i] for i in topic.argsort()[-30:]])
    print('\n')
    sub_list.append([vectorizer.get_feature_names()[i] for i in topic.argsort()[-30:]])
    word_list.append(sub_list)

word_list

df_topic['Topic']=[1,2,3,4,5,6,7,8,9,10]
df_topic['TopWords']=word_list

df_topic.head()
df_topic.to_csv('../output/topic top words v2.csv',index=None)
```

```
/Users/aprilyang/opt/anaconda3/lib/python3.8/site-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run
```

```
_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argu
ment and any exception that happen during thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
Top 30 words for topic #0:
['wait', 'teamtrump', 'service', 'building', 'press', 'million', 'war', 'energy', 'truly', 'dont', 'lot', 'candidat
e', 'remember', 'open', 'presidential', 'end', 'problem', 'live', 'place', 'soon', 'doesnt', 'white', 'nation', 'whit
ehouse', 'sta', 'people', 'better', 'watch', 'look', 'house']


Top 30 words for topic #1:
['absolutely', 'highest', 'given', 'robe', 'mark', 'miss', 'save', 'celebrity', 'win', 'russian', 'apprenticenbc', 'g
od', 'wow', 'ivankatrump', 'happen', 'stand', 'celebapprentice', 'apprentice', 'justice', 'case', 'senator', 'cou',
'witch', 'hunt', 'rating', 'book', 'congratulation', 'tonight', 'best', 'great']


Top 30 words for topic #2:
['failing', 'success', 'texas', 'going', 'price', 'south', 'truth', 'cruz', 'tariff', 'drug', 'korea', 'federal', 'co
mpany', 'lie', 'schiff', 'lost', 'happy', 'york', 'hit', 'fantastic', 'iran', 'time', 'course', 'wall', 'border', 'fb
i', 'security', 'story', 'record', 'national']


Top 30 words for topic #3:
['election', 'history', 'administration', 'didnt', 'far', 'dont', 'thats', 'best', 'usa', 'republican', 'russia', 'ba
d', 'right', 'night', 'said', 'impeachment', 'say', 'working', 'real', 'hard', 'democrat', 'foxnews', 'united', 'worl
d', 'win', 'really', 'state', 'country', 'people', 'job']


Top 30 words for topic #4:
['tower', 'criminal', 'sign', 'allowed', 'stock', 'comey', 'ive', 'voting', 'future', 'hotel', 'ready', 'market', 'po
tus', 'rally', 'government', 'golf', 'florida', 'john', 'makeamericagreatagain', 'billion', 'tomorrow', 'forward', 'm
an', 'people', 'got', 'work', 'make', 'donald', 'america', 'great']


Top 30 words for topic #5:
['fraud', 'force', 'southern', 'phony', 'country', 'stay', 'ing', 'jim', 'disaster', 'voter', 'iowa', 'major', 'fox',
'dems', 'point', 'number', 'office', 'honor', 'interview', 'stop', 'election', 'democrat', 'let', 'poll', 'republica
n', 'vote', 'repo', 'medium', 'fake', 'news']


Top 30 words for topic #6:
['year', 'guy', 'carolina', 'city', 'million', 'political', 'ant', 'order', 'long', 'making', 'illegal', 'wonderful',
'governor', 'woman', 'state', 'coming', 'crooked', 'maga', 'impo', 'life', 'looking', 'law', 'campaign', 'clinton',
'country', 'love', 'hillary', 'obama', 'great', 'american']
```

```
Top 30 words for topic #7:
['depa', 'crazy', 'bad', 'donaldjtrumpjr', 'mean', 'believe', 'night', 'race', 'fighting', 'taking', 'entrepreneur',
'rate', 'seen', 'month', 'ago', 'tremendous', 'sad', 'friend', 'home', 'trying', 'word', 'change', 'join', 'mexico',
'debate', 'thing', 'cnn', 'going', 'true', 'year']


Top 30 words for topic #8:
['politics', 'information', 'special', 'highly', 'approval', 'investigation', 'loser', 'used', 'interviewed', 'bush',
'youre', 'cont', 'told', 'joe', 'mike', 'general', 'corrupt', 'person', 'agree', 'team', 'trump', 'yesterday', 'muell
er', 'enjoy', 'obamacare', 'morning', 'collusion', 'meeting', 'getting', 'foxandfriends']


Top 30 words for topic #9:
['biden', 'dollar', 'sma', 'money', 'gop', 'leader', 'make', 'trade', 'week', 'crime', 'senate', 'america', 'amazin
g', 'congress', 'fact', 'family', 'economy', 'business', 'strong', 'military', 'total', 'vote', 'democrat', 'tax', 'b
order', 'suppo', 'china', 'deal', 'run', 'need']
```

In [19]:

```python
topic_values = lda_model.transform(data_matrix)
tweets_df['Topic'] = topic_values.argmax(axis=1)
tweets_df.head()
```

```
/Users/aprilyang/opt/anaconda3/lib/python3.8/site-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run
_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argu
ment and any exception that happen during thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

Out[19]:

	text	Topic
0	Republicans and Democrats have both created ou...	0
1	I was thrilled to be back in the Great city of...	6
2	RT @CBS_Herridge: READ: Letter to surveillance...	6
3	The Unsolicited Mail In Ballot Scam is a major...	5
4	RT @MZHemingway: Very friendly telling of even...	9

In [20]:

```python
# outputting the topic label and writing into a new csv file
df['Topic'] = topic_values.argmax(axis=1)
df.to_csv('../output/tweets_with_topic_label.csv',index=None)
```

```
/Users/aprilyang/opt/anaconda3/lib/python3.8/site-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run
```

```
_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argu
ment and any exception that happen during thetransform in `preprocessing_exc_tuple` in IPython 7.17 and above.
  and should_run_async(code)
```

In [ ]:

```

```

In [2]:

```
# !pip install vaderSentiment
# !pip install -U textblob
# !pip install swifter
!pip install gensim==3.8.3
```

```
Requirement already satisfied: gensim==3.8.3 in /Users/my_love/opt/anaconda3/lib/python3.8/site-packages (3.8.3)
Requirement already satisfied: scipy>=0.18.1 in /Users/my_love/opt/anaconda3/lib/python3.8/site-packages (from gensim
==3.8.3) (1.5.2)
Requirement already satisfied: smart-open>=1.8.1 in /Users/my_love/opt/anaconda3/lib/python3.8/site-packages (from ge
nsim==3.8.3) (5.0.0)
Requirement already satisfied: six>=1.5.0 in /Users/my_love/opt/anaconda3/lib/python3.8/site-packages (from gensim==
3.8.3) (1.15.0)
Requirement already satisfied: numpy>=1.11.3 in /Users/my_love/opt/anaconda3/lib/python3.8/site-packages (from gensim
==3.8.3) (1.19.2)
```

In [3]:

```
from nltk.corpus import stopwords
import matplotlib.pyplot as plt
from textblob import TextBlob
import seaborn as sns
from tqdm import tqdm
from time import time
import pandas as pd
import numpy as np
import regex as re
import unicodedata
import swifter
import gc
import json

import nltk
sns.set_style('white')

from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from sklearn.feature_extraction.text import TfidfVectorizer
from gensim.models.phrases import Phrases, Phraser
from nltk.tokenize import TweetTokenizer
from gensim.models import Word2Vec
from sklearn.cluster import KMeans

from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, recall_score, f1_score
```

In [4]:
```
df = pd.read_csv('../data/tweets_01-08-2021.csv')
df
```

Out[4]:

	id	text	isRetweet	isDeleted	device	favorites	retweets	date	isFlagged
0	98454970654916608	Republicans and Democrats have both created ou...	f	f	TweetDeck	49	255	2011-08-02 18:07:48	f
1	1234653427789070336	I was thrilled to be back in the Great city of...	f	f	Twitter for iPhone	73748	17404	2020-03-03 01:34:50	f
2	1218010753434820614	RT @CBS_Herridge: READ: Letter to surveillance...	t	f	Twitter for iPhone	0	7396	2020-01-17 03:22:47	f
3	1304875170860015617	The Unsolicited Mail In Ballot Scam is a major...	f	f	Twitter for iPhone	80527	23502	2020-09-12 20:10:58	f
4	1218159531554897920	RT @MZHemingway: Very friendly telling of even...	t	f	Twitter for iPhone	0	9081	2020-01-17 13:13:59	f
...	...	...	...	...	...	...	...	...	...
56566	1319485303363571714	RT @RandPaul: I don't know why @JoeBiden think...	t	f	Twitter for iPhone	0	20683	2020-10-23 03:46:25	f
56567	1319484210101379072	RT @EliseStefanik: President @realDonaldTrump ...	t	f	Twitter for iPhone	0	9869	2020-10-23 03:42:05	f
56568	1319444420861829121	RT @TeamTrump: LIVE: Presidential Debate #Deba...	t	f	Twitter for iPhone	0	8197	2020-10-23 01:03:58	f
56569	1319384118849949702	Just signed an order to support the workers of...	f	f	Twitter for iPhone	176289	36001	2020-10-22 21:04:21	f
56570	1319345719829008387	Suburban women want Safety &amp; Security. Joe...	f	f	Twitter for iPhone	95169	19545	2020-10-22 18:31:46	f

56571 rows × 9 columns

Sentiment Analysis

Pre-processing the data

Source: https://link-springer-com.ezproxy.cul.columbia.edu/content/pdf/10.1007%2F978-3-319-09339-0.pdf (page 617)

In [5]:

```
# Add additional feature 'retweeted'
tweets = df['text'].to_list()
values = []

for tweet in tweets:
    if tweet.startswith('RT'):
        value = True
    else:
        value = False

    values.append(value)

df['retweeted'] = values
```

In [6]:

```
# Step 1: Denoising – Remove Username, Hashtags, Links, Change to lowercase
def denoise(tweets):

    clean_tweets = []

    for tweet in tweets:
        result = unicodedata.normalize('NFKD', tweet)
        result = re.sub("@(\w{1,15})", " ", result) # mentions
        result = re.sub("#(\w{1,15})", " ", result) # hashtags
        result = re.sub("https?://([^\s]+)", ' ', result) # links
        result = re.sub("RT", ' ', result) # RT :

        result = re.sub(" &amp", ' ', result) # &amp
        result = re.sub("[\n\r\t\0]", ' ', result) # new line, tabs, etc

        result = re.sub(r"\'t", "not", result)
        result = re.sub(r"\'re", " are", result)
        result = re.sub(r"\'s", " is", result)
        result = re.sub(r"\'d", " would", result)
        result = re.sub(r"\'ll", " will", result)
        result = re.sub(r"\'ve", " have", result)
        result = re.sub(r"\'m", " am", result)

        result = re.sub(r'\b\w\b', ' ', result) # sigle letter
        result = re.sub('[!,.-;:\+\-\()?"“”\[\]{}]', ' ', result) # punct

        result = re.sub('\s{2,}', ' ', result) # 2+ whitespaces
        result = result.strip()

        clean_tweets.append(result)

    return clean_tweets
```

In [7]:

```
# Step 2: Normalizing contractions
# source: https://towardsdatascience.com/text-normalization-7ecc8e084e31

def normalize_contractions(tweets):
    contraction_list = json.loads(open('../data/english_contractions.json', 'r').read())
    clean_tweets = []

    for tweet in tweets:
        clean_tweets.append(_normalize_contractions_text(tweet, contraction_list))

    return clean_tweets

def _normalize_contractions_text(text, contractions):
    """
    This function normalizes english contractions.
    """
    new_token_list = []
    token_list = text.split()

    for word_pos in range(len(token_list)):
        word = token_list[word_pos]
        first_upper = False
        if word[0].isupper():
            first_upper = True
        if word.lower() in contractions:
            replacement = contractions[word.lower()]
            if first_upper:
                replacement = replacement[0].upper()+replacement[1:]
            replacement_tokens = replacement.split()
            if len(replacement_tokens)>1:
                new_token_list.append(replacement_tokens[0])
                new_token_list.append(replacement_tokens[1])
            else:
                new_token_list.append(replacement_tokens[0])
        else:
            new_token_list.append(word)

    tweet = " ".join(new_token_list).strip(" ").lower()

    return tweet
```

In [8]:

```
def remove_stop_words(tweets):
    stopwords_english = stopwords.words('english')

    to_be_removed = ["haven't", "against", "not", "weren't", "won't", 'no']

    for word in to_be_removed:
        stopwords_english.remove(word)

    stopwords_english.append('pm')
    stopwords_english.append('am')

    clean_tweets = []

    # instantiate the tokenizer class
    tokenizer = TweetTokenizer(preserve_case=False,
                               strip_handles=True,
                               reduce_len=True)

    for tweet in tweets:

        # tokenize the tweets
        tweet_tokens = tokenizer.tokenize(tweet)

        tweet_clean = ''

        for word in tweet_tokens: # Go through every word in your tokens list
            if word not in stopwords_english:
                tweet_clean = tweet_clean + ' ' + word

        clean_tweets.append(tweet_clean.strip())

    return clean_tweets
```

In [9]:

```
def deEmojify(tweets):
    clean_tweets = []
    for tweet in tweets:
        regrex_pattern = re.compile(pattern = "["
            u"\U0001F600-\U0001F64F"  # emoticons
            u"\U0001F300-\U0001F5FF"  # symbols & pictographs
            u"\U0001F680-\U0001F6FF"  # transport & map symbols
            u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                               "]+", flags = re.UNICODE)

        clean_tweets.append(regrex_pattern.sub(r'', tweet))

    return clean_tweets
```

In [10]:
```
def normalization_pipeline(tweets):

    tweets = denoise(tweets)
    tweets = deEmojify(tweets)
    tweets = normalize_contractions(tweets)
    tweets = remove_stop_words(tweets)

    return tweets
```

In [11]:
```
# load tweets
tweets = df['text'].to_list()

clean_tweets = normalization_pipeline(tweets)
```

In [12]:
```
df['sentiment_text'] = clean_tweets
```

In [13]:
```
#remove tweets that are empty
df = df[df['sentiment_text'] != '']
```

Sentiment Analysis

Part 1: Vader & TextBlob

In [14]:
```
def sentiment_scores(tweet):
    # Create a SentimentIntensityAnalyzer object.
    sid_obj = SentimentIntensityAnalyzer()

    sentiment_score = sid_obj.polarity_scores(tweet)['compound']
    blob_dict = TextBlob(tweet).sentiment

    sentiment_vader.append(sentiment_score)
    polarity.append(blob_dict.polarity)
    subjectivity.append(blob_dict.subjectivity)
```

In [15]:

```
# compute scores
tweets_clean = df['sentiment_text']

polarity = []
subjectivity = []
sentiment_vader = []

for tweet in tweets_clean:
    sentiment_scores(tweet)

df['subjectivity_score'] = subjectivity
df['TextBlob_sa'] = polarity
df['Vader_sa'] = sentiment_vader
```

```
<ipython-input-15-aad1d5f1801e>:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  df['subjectivity_score'] = subjectivity
<ipython-input-15-aad1d5f1801e>:12: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  df['TextBlob_sa'] = polarity
<ipython-input-15-aad1d5f1801e>:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
```

```
-a-view-versus-a-copy
  df['Vader_sa'] = sentiment_vader
```

Plotting the results

In [16]:

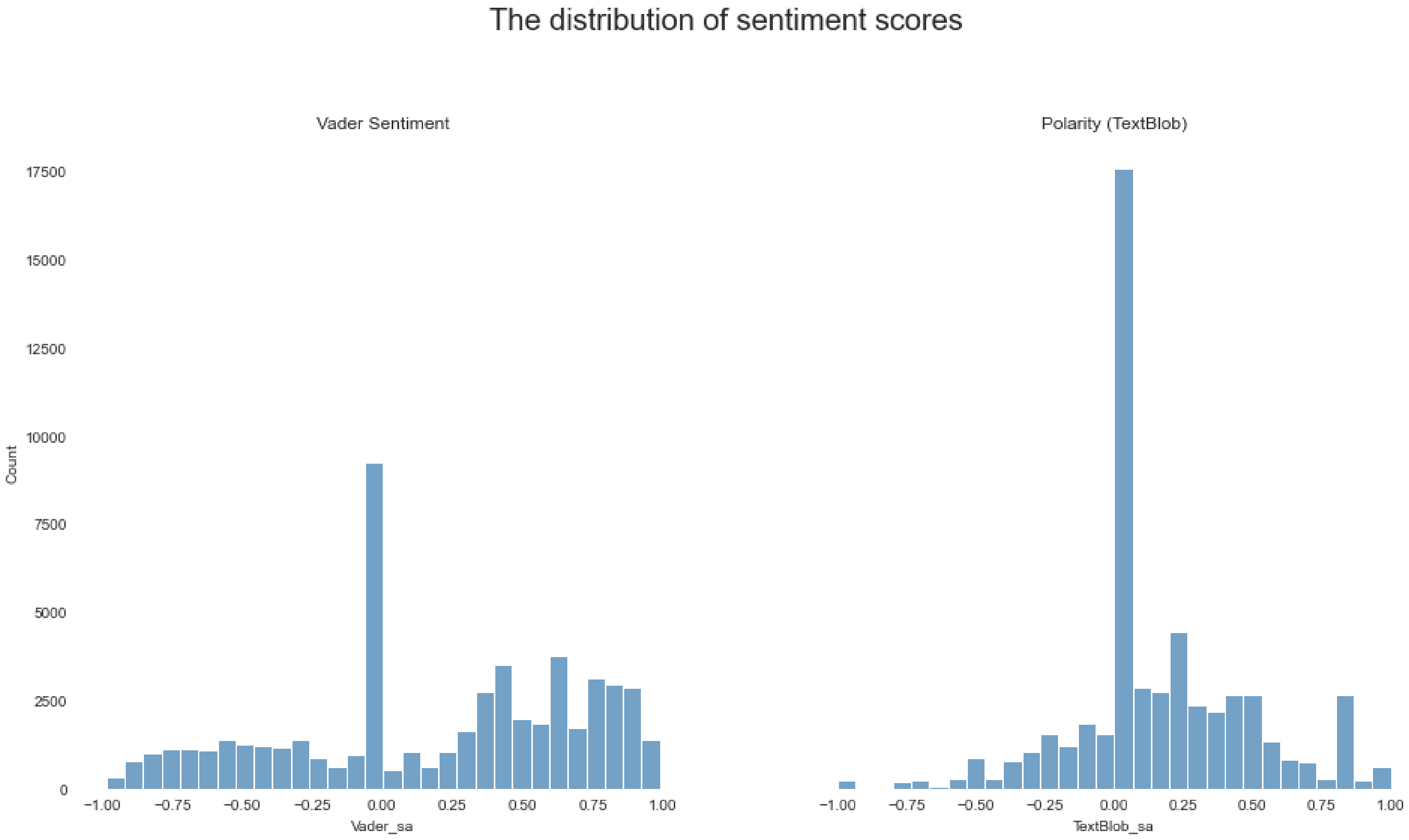
```python
def get_rid_of_spine(axes):
    for ax in axes:
        for spine in ax.spines.values():
            spine.set_visible(False)
```

In [17]:

```python
# Plotting polarity vs sentiment
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16,8), sharey=True)
g1 = sns.histplot(x=df['Vader_sa'], color="steelblue", ax=ax1, bins = 30)
g2 = sns.histplot(x=df['TextBlob_sa'], color="steelblue", ax=ax2, bins = 30)

ax1.set_title("Vader Sentiment")
ax2.set_title("Polarity (TextBlob)")

get_rid_of_spine([ax1, ax2])
plt.suptitle("The distribution of sentiment scores", y = 1.03, fontsize=20)
```

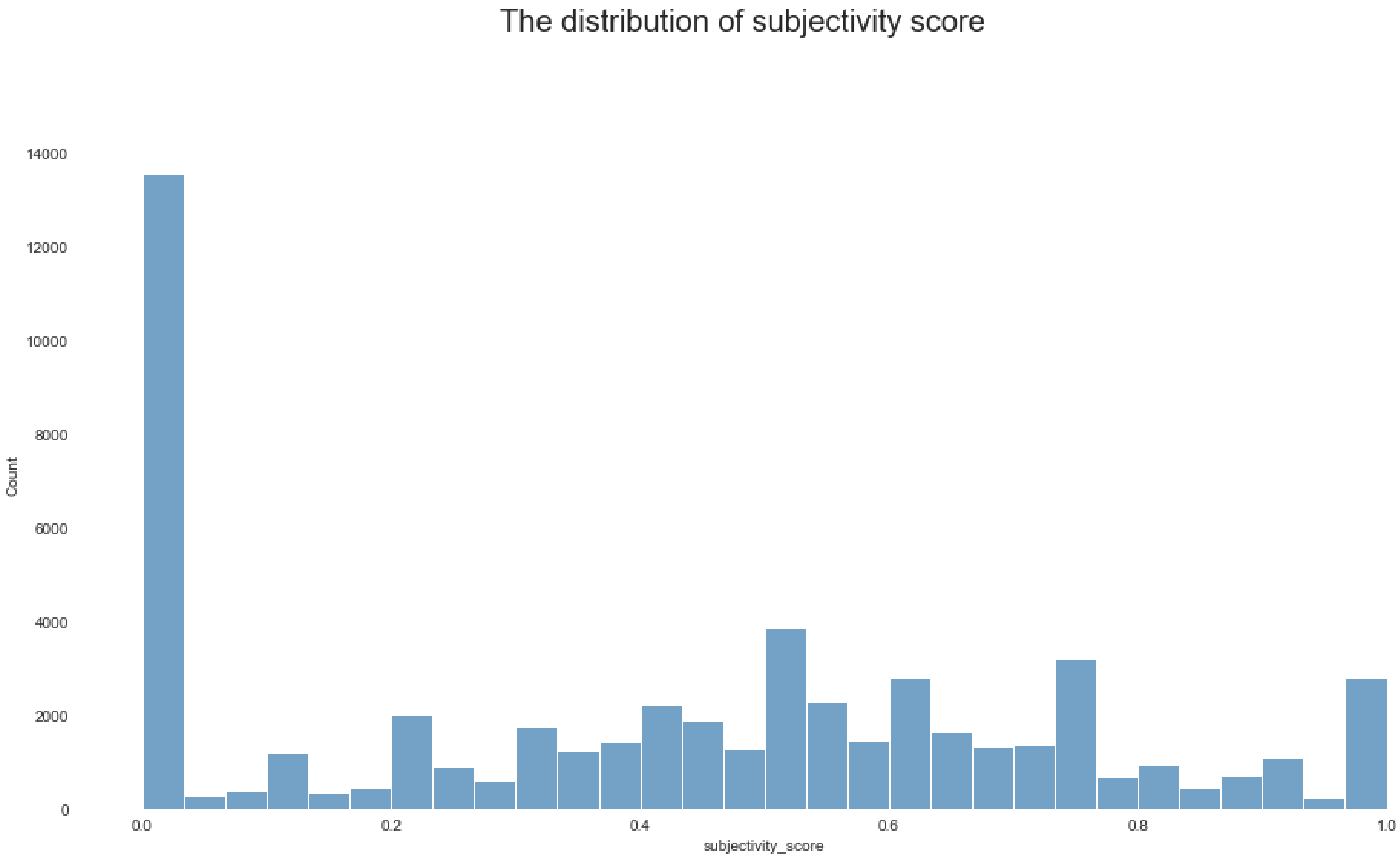
Out[17]:

```
Text(0.5, 1.03, 'The distribution of sentiment scores')
```



```
In [18]: # Plotting objectivity
         # subjectivity_score = 0: very objective
         # subjectivity_score = 1: very subjective
         fig, ax = plt.subplots(figsize=(16,8))
         g1 = sns.histplot(x=df['subjectivity_score'], color="steelblue", bins = 30)
         get_rid_of_spine([ax])
         plt.suptitle("The distribution of subjectivity score", y = 1.03, fontsize=20)
```

```
Out[18]: Text(0.5, 1.03, 'The distribution of subjectivity score')
```



```
In [19]: df.to_csv(r'../output/sentiment_analysis_clean.csv', index = False)
```

Part 2: Word-2-vec and KMeans

Source: https://towardsdatascience.com/unsupervised-sentiment-analysis-a38bf1906483

In [20]:

```
# loading clean dataset
df_tweets = pd.read_csv('../output/sentiment_analysis_clean.csv')
df_tweets.head()
```

Out[20]:

	id	text	isRetweet	isDeleted	device	favorites	retweets	date	isFlagged	retweeted	sentiment_text	subj
0	98454970654916608	Republicans and Democrats have both created ou...	f	f	TweetDeck	49	255	2011-08-02 18:07:48	f	False	republicans democrats created economic problems	
1	1234653427789070336	I was thrilled to be back in the Great city of...	f	f	Twitter for iPhone	73748	17404	2020-03-03 01:34:50	f	False	thrilled back great city charlotte north carol...	
2	1218010753434820614	RT @CBS_Herridge: READ: Letter to surveillance...	t	f	Twitter for iPhone	0	7396	2020-01-17 03:22:47	f	True	read letter surveillance court obtained cbs ne...	
3	1304875170860015617	The Unsolicited Mail In Ballot Scam is a major...	f	f	Twitter for iPhone	80527	23502	2020-09-12 20:10:58	f	False	unsolicited mail ballot scam major threat demo...	
4	1218159531554897920	RT @MZHemingway: Very friendly telling of even...	t	f	Twitter for iPhone	0	9081	2020-01-17 13:13:59	f	True	friendly telling events comey apparent leaking...	

In [21]:

```
# Removing empty sentiment_text
df_tweets = df_tweets[-df_tweets.sentiment_text.isna()]

# Tokenizing sentiment_text for compatibility with gensim package
df_tweets.sentiment_text = df_tweets.sentiment_text.swifter.apply(lambda x: x.split())

df_tweets.sentiment_text.head()
```

```
Out[21]: 0    [republicans, democrats, created, economic, pr...
         1    [thrilled, back, great, city, charlotte, north...
         2    [read, letter, surveillance, court, obtained, ...
         3    [unsolicited, mail, ballot, scam, major, threa...
         4    [friendly, telling, events, comey, apparent, l...
         Name: sentiment_text, dtype: object
```

```
In [22]: corpus = [tweet for tweet in df_tweets.sentiment_text]
         phrases = Phrases(corpus, min_count=10)
         bigram = Phraser(phrases)
         sentences = bigram[corpus]
```

```
In [23]: # Example of sentence with bigram token
         sentences[1]
```

```
Out[23]: ['thrilled_back',
          'great',
          'city',
          'charlotte',
          'north_carolina',
          'thousands',
          'hardworking_american',
          'patriots',
          'love',
          'country',
          'cherish',
          'values',
          'respect',
          'laws',
          'always',
          'put',
          'america',
          'first',
          'thank',
          'wonderful',
          'evening']
```

```
In [24]: w2v_model = Word2Vec(min_count=1,
                              window=7,
                              size=300,
                              sample=1e-5,
                              alpha=0.03,
                              min_alpha=0.0007,
                     negative=10,
                     workers=-1)

start = time()

w2v_model.build_vocab(sentences,
                      progress_per=50000)

print('Time to build vocab: {} mins'.format(round((time() - start) / 60, 2)))
```

```
Time to build vocab: 0.1 mins
```

In [25]:

```
start = time()

w2v_model.train(sentences, total_examples=w2v_model.corpus_count, epochs=200, report_delay=1)

print('Time to train the model: {} mins'.format(round((time() - start) / 60, 2)))

w2v_model.init_sims(replace=True)
```

```
Time to train the model: 3.03 mins
```

In [27]:

```
#w2v_model.save("../output/word2vec.model")

word_vectors = Word2Vec.load("../output/word2vec.model").wv
```

In [28]:

```
# build Kmeans model
model = KMeans(n_clusters=2, max_iter=1000, random_state=True, n_init=50).fit(X=word_vectors.vectors.astype('double')
word_vectors.similar_by_vector(model.cluster_centers_[0], topn=50, restrict_vocab=None)
```

Out[28]:

```
[('guilt', 0.23333770036697388),
 ('lots_money', 0.2269250452518463),
 ('alec', 0.22211502492427826),
 ('shadows', 0.22118237614631653),
 ('fact', 0.21808487176895142),
 ('buttigieg', 0.21547682583332062),
 ('riptides', 0.21463802456855774),
 ('machine', 0.21460095047950745),
 ('mayer', 0.2124587893486023),
 ('registered', 0.20803800225257874),
 ('bcuz', 0.20577329397201538),
 ('towns', 0.2036629617214203),
 ('viewership', 0.20263740420341492),
 ('flotus', 0.20160475373268127),
 ('defunds', 0.20114798843860626),
 ('الولايات', 0.19775941967964172),
 ('wind', 0.1967260241508484),
 ('commenting', 0.1965508908033371),
 ('gifting', 0.18973805010318756),
 ('ymdh', 0.18771769106388092),
 ('crazy_bernie', 0.1873263716697693),
 ('authorized', 0.1855493187904358),
 ('hollywood', 0.18417152762413025),
 ('minnis', 0.1813950091600418),
 ('que', 0.18110191822052002),
 ('connect', 0.17977237701416016),
 ('extort', 0.17782193422317505),
 ('pgimqykpoj', 0.17758050560951233),
 ('forthcoming', 0.1773029863834381),
 ('citizenwhere', 0.1770586222410202),
 ('tulsi', 0.1764734536409378),
 ('supercuts', 0.1760212182998657 2),
 ('shinzo', 0.17547494173049927),
 ('unusable', 0.17516516149044037),
 ('large', 0.1751020848751068),
 ('greeted', 0.17487813532352448),
 ('islambies', 0.17445221543312073),
 ('consists', 0.17433936893939972),
 ('captivates', 0.17376179993152618),
 ('outsmarts', 0.1736268550157547),
 ('georgians', 0.17339861392974854),
 ('enforcer', 0.17309720814228058),
 ('incontrovertibl', 0.17299871146678925),
 ('disparity', 0.17234468460083008),
 ('jailed', 0.1722993552684784),
 ('brokering', 0.17192861437797546),
 ('enacted', 0.1718808263540268),
 ('naturall', 0.17064031958580017),
 ('showed', 0.17052684724330902),
 ('apartments', 0.17025959491729736)]
```

In [31]:

```
positive_cluster_index = 1
positive_cluster_center = model.cluster_centers_[positive_cluster_index]
negative_cluster_center = model.cluster_centers_[1-positive_cluster_index]

words = pd.DataFrame(word_vectors.vocab.keys())
words.columns = ['words']
```

```
words['vectors'] = words.words.apply(lambda x: word_vectors[f'{x}'])
words['cluster'] = words.vectors.apply(lambda x: model.predict([np.array(x)]))
words.cluster = words.cluster.apply(lambda x: x[0])

words['cluster_value'] = [1 if i==positive_cluster_index else -1 for i in words.cluster]
words['closeness_score'] = words.apply(lambda x: 1/(model.transform([x.vectors]).min()), axis=1)
words['sentiment_coeff'] = words.closeness_score * words.cluster_value

words.head(20)
```

Out[31]:

	words	vectors	cluster	cluster_value	closeness_score	sentiment_coeff
0	republicans	[-0.05263862, -0.008828307, 0.040902816, -0.06...	0	-1	0.999885	-0.999885
1	democrats	[0.08512214, 0.009598418, -0.030342635, -0.084...	0	-1	1.000099	-1.000099
2	created	[0.05602551, 0.077005245, -0.010651855, -0.064...	1	1	0.998899	0.998899
3	economic	[-0.009064345, 0.04958363, -0.059036087, -0.03...	0	-1	1.004524	-1.004524
4	problems	[0.044180546, -0.05363604, -0.062909886, 0.002...	0	-1	1.001660	-1.001660
5	thrilled_back	[0.054495912, -0.071668714, -0.0046636625, 0.0...	0	-1	1.001481	-1.001481
6	great	[0.033503465, -0.067113996, 0.0569714, -0.0743...	0	-1	0.999776	-0.999776
7	city	[-0.053522754, -0.06895677, -0.063024454, 0.02...	0	-1	1.001558	-1.001558
8	charlotte	[-0.029565733, -0.042288236, 0.047731206, 0.09...	0	-1	1.000492	-1.000492
9	north_carolina	[0.038449172, -0.015750559, 0.056030795, 0.054...	0	-1	1.003523	-1.003523
10	thousands	[0.04322429, -0.046556123, 0.07218951, 0.01578...	0	-1	1.001836	-1.001836
11	hardworking_american	[-0.081599146, -0.093259536, 0.012313393, -0.0...	0	-1	0.998893	-0.998893
12	patriots	[0.02682546, -0.08673129, 0.03174281, -0.01708...	0	-1	1.004781	-1.004781
13	love	[0.006338382, -0.040629767, -0.041294783, -0.0...	0	-1	0.999106	-0.999106
14	country	[-0.05210093, -0.032660868, -0.047626328, -0.0...	1	1	1.001930	1.001930
15	cherish	[-0.044070672, 0.009197703, 0.0018399832, -0.0...	0	-1	1.005497	-1.005497
16	values	[0.06377718, -0.044265706, 0.039525192, 0.0531...	1	1	1.000060	1.000060
17	respect	[-0.092620395, -0.008787945, -0.087396756, -0....	0	-1	0.999891	-0.999891
18	laws	[-0.073388085, -0.085784644, 0.08755252, -0.04...	1	1	0.999209	0.999209

	words	vectors	cluster	cluster_value	closeness_score	sentiment_coeff
19	always	[0.06472598, -0.09232594, 0.08492987, 0.056004...	0	-1	1.003001	-1.003001

In [32]:

```python
words[['words', 'sentiment_coeff']].to_csv('../output/sentiment_dictionary.csv', index=False)
```

In [33]:

```python
df_tweets.sentiment_text = df_tweets.sentiment_text.swifter.apply(lambda x: ' '.join(bigram[x]))

cut_labels = [-1, 0, 1]
cut_bins = [-1, -0.00000001, 0.00000001, 1]
df_tweets['VADER'] = pd.cut(df_tweets.Vader_sa, bins=cut_bins, labels=cut_labels)
df_tweets['TextBlob'] = pd.cut(df_tweets.TextBlob_sa, bins=cut_bins, labels=cut_labels)

df_tweets['VADER'].value_counts()
```

Out[33]:

```
1     31055
-1    14920
0      8699
Name: VADER, dtype: int64
```

In [34]:

```python
df_tweets.head()
```

Out[34]:

	id	text	isRetweet	isDeleted	device	favorites	retweets	date	isFlagged	retweeted	sentiment_text	subj
0	98454970654916608	Republicans and Democrats have both created ou...	f	f	TweetDeck	49	255	2011-08-02 18:07:48	f	False	republicans democrats created economic problems	
1	1234653427789070336	I was thrilled to be back in the Great city of...	f	f	Twitter for iPhone	73748	17404	2020-03-03 01:34:50	f	False	thrilled_back great city charlotte north_carol...	
2	1218010753434820614	RT @CBS_Herridge: READ: Letter to surveillance...	t	f	Twitter for iPhone	0	7396	2020-01-17 03:22:47	f	True	read letter surveillance court obtained cbs ne...	

	id	text	isRetweet	isDeleted	device	favorites	retweets	date	isFlagged	retweeted	sentiment_text	subj
3	1304875170860015617	The Unsolicited Mail In Ballot Scam is a major...	f	f	Twitter for iPhone	80527	23502	2020-09-12 20:10:58	f	False	unsolicited mail_ballot scam major threat demo...	
4	1218159531554897920	RT @MZHemingway: Very friendly telling of even...	t	f	Twitter for iPhone	0	9081	2020-01-17 13:13:59	f	True	friendly telling events comey apparent leaking...	

In [35]:

```
df_tweets[['sentiment_text', 'VADER', 'TextBlob']].to_csv('../output/cleaned_dataset.csv', index=False)
```

In [36]:

```
final_file = pd.read_csv('../output/cleaned_dataset.csv')
sentiment_map = pd.read_csv('../output/sentiment_dictionary.csv')
sentiment_dict = dict(zip(sentiment_map.words.values, sentiment_map.sentiment_coeff.values))
file_weighting = final_file.copy()
```

In [37]:

```
tfidf = TfidfVectorizer(tokenizer=lambda y: y.split(), norm=None)
tfidf.fit(file_weighting.sentiment_text)
features = pd.Series(tfidf.get_feature_names())
transformed = tfidf.transform(file_weighting.sentiment_text)
```

```
/Users/my_love/opt/anaconda3/lib/python3.8/site-packages/sklearn/feature_extraction/text.py:489: UserWarning: The par
ameter 'token_pattern' will not be used since 'tokenizer' is not None'
  warnings.warn("The parameter 'token_pattern' will not be used"
```

In [38]:

```
def create_tfidf_dictionary(x, transformed_file, features):
    '''
    create dictionary for each input sentence x, where each word has assigned its tfidf score

    inspired  by function from this wonderful article:
    https://medium.com/analytics-vidhya/automated-keyword-extraction-from-articles-using-nlp-bfd864f41b34

    x - row of dataframe, containing sentences, and their indexes,
    transformed_file - all sentences transformed with TfidfVectorizer
    features - names of all words in corpus used in TfidfVectorizer

    '''
    vector_coo = transformed_file[x.name].tocoo()
    vector_coo.col = features.iloc[vector_coo.col].values
    dict_from_coo = dict(zip(vector_coo.col, vector_coo.data))
    return dict_from_coo

def replace_tfidf_words(x, transformed_file, features):
    '''
    replacing each word with it's calculated tfidf dictionary with scores of each word
    x - row of dataframe, containing sentences, and their indexes,
    transformed_file - all sentences transformed with TfidfVectorizer
    features - names of all words in corpus used in TfidfVectorizer
    '''
    dictionary = create_tfidf_dictionary(x, transformed_file, features)
    return list(map(lambda y:dictionary[f'{y}'], x.sentiment_text.split()))
```

In [39]:

```
%%time
replaced_tfidf_scores = file_weighting.apply(lambda x: replace_tfidf_words(x, transformed, features), axis=1)
#this step takes around 3-4 minutes minutes to calculate
```

```
CPU times: user 12.6 s, sys: 41.4 ms, total: 12.7 s
Wall time: 12.7 s
```

In [40]:

```
def replace_sentiment_words(word, sentiment_dict):
    '''
    replacing each word with its associated sentiment score from sentiment dict
    '''
    try:
        out = sentiment_dict[word]
    except KeyError:
        out = 0
    return out
```

In [41]:

```
replaced_closeness_scores = file_weighting.sentiment_text.apply(lambda x: list(map(lambda y: replace_sentiment_words
```

In [42]:

```
replacement_df = pd.DataFrame(data=[replaced_closeness_scores, replaced_tfidf_scores, file_weighting.sentiment_text,
replacement_df.columns = ['sentiment_coeff', 'tfidf_scores', 'sentence', 'sentiment']
replacement_df['sentiment_rate'] = replacement_df.apply(lambda x: np.array(x.loc['sentiment_coeff']) @ np.array(x.loc
replacement_df['prediction'] = (replacement_df.sentiment_rate>0).astype('int8')
replacement_df['sentiment'] = [1 if i==1 else -1 for i in replacement_df.sentiment]
replacement_df['prediction'].replace(0,-1, inplace=True)
```

In [43]:

```
replacement_df.head()
```

Out[43]:

	sentiment_coeff	tfidf_scores	sentence	sentiment	sentiment_rate	prediction
0	[-0.999885146975044, -1.0000988354469655, 0.99...	[5.489166917397826, 4.58797328880549, 7.284189...	republicans democrats created economic problems	-1	-15.977217	-1
1	[-1.0014808463847191, -0.9997763344558817, -1....	[9.075948501488753, 3.1859305707174603, 6.2960...	thrilled_back great city charlotte north_carol...	1	-52.411713	-1
2	[1.0005741629267897, 0.99923881523924, -0.9997...	[6.166158657735486, 7.565356423691285, 8.38280...	read letter surveillance court obtained cbs ne...	-1	-13.590002	-1
3	[-0.9994979770646428, 1.0023169213337315, 1.00...	[9.511266572746598, 8.913429571990978, 7.00388...	unsolicited mail_ballot scam major threat demo...	-1	-55.314454	-1
4	[-1.0005131149446047, -0.9996004337087409, 1.0...	[8.61332497954064, 7.632495726528913, 7.689654...	friendly telling events comey apparent leaking...	1	-0.852499	-1

In [44]:

```
replacement_df.prediction.value_counts()
```

Out[44]:

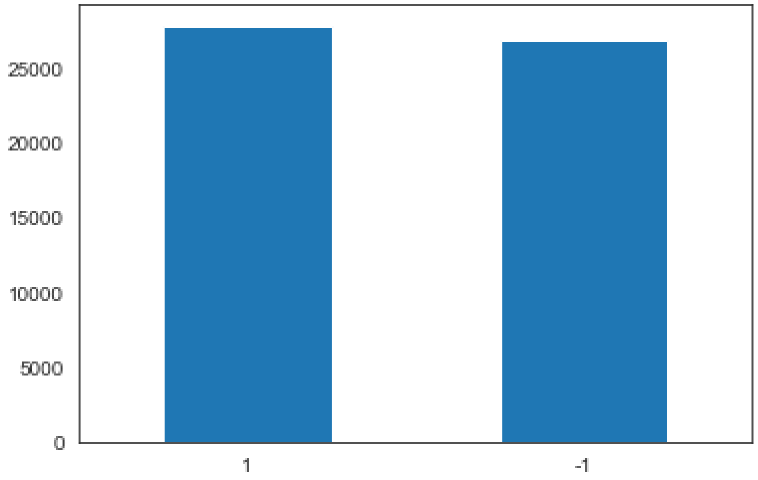
```
 1    27807
-1    26867
Name: prediction, dtype: int64
```

In [45]:

```
replacement_df.prediction.value_counts().plot(kind='bar', rot=0)
```

Out[45]:

```
<AxesSubplot:>
```



In [46]:
```
df_merged = pd.concat([df_tweets, replacement_df], axis=1)
df_merged.rename(columns={"prediction": "W2V-kNN"}, inplace=True)
df_merged.columns
```

Out[46]:
```
Index(['id', 'text', 'isRetweet', 'isDeleted', 'device', 'favorites',
       'retweets', 'date', 'isFlagged', 'retweeted', 'sentiment_text',
       'subjectivity_score', 'TextBlob_sa', 'Vader_sa', 'VADER', 'TextBlob',
       'sentiment_coeff', 'tfidf_scores', 'sentence', 'sentiment',
       'sentiment_rate', 'W2V-kNN'],
      dtype='object')
```

In [47]:
```
df_final = df_merged.iloc[:,[0,1,10,11,14,15,-1]]
df_final.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 54674 entries, 0 to 54673
Data columns (total 7 columns):
 #   Column              Non-Null Count  Dtype   
---  ------              --------------  -----   
 0   id                  54674 non-null  int64   
 1   text                54674 non-null  object  
 2   sentiment_text      54674 non-null  object  
 3   subjectivity_score  54674 non-null  float64 
 4   VADER               54674 non-null  category
 5   TextBlob            54422 non-null  category
 6   W2V-kNN             54674 non-null  int8    
```

```
dtypes: category(2), float64(1), int64(1), int8(1), object(2)
memory usage: 4.7+ MB
```

In [48]:

```
df_final.dropna(how='any', inplace=True)
df_final.VADER = df_final.VADER.astype(int)
df_final.TextBlob = df_final.TextBlob.cat.codes - 1
df_final['W2V-kNN'] = df_final['W2V-kNN'].astype(int)
```

```
<ipython-input-48-bdbf950dc954>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  df_final.dropna(how='any', inplace=True)
/Users/my_love/opt/anaconda3/lib/python3.8/site-packages/pandas/core/generic.py:5168: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  self[name] = value
<ipython-input-48-bdbf950dc954>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  df_final['W2V-kNN'] = df_final['W2V-kNN'].astype(int)
```

In [49]:

```
df_final['Final'] = (df_final.VADER + df_final.TextBlob + df_final['W2V-kNN'])/3
```

```
<ipython-input-49-62f24c0e67c7>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  df_final['Final'] = (df_final.VADER + df_final.TextBlob + df_final['W2V-kNN'])/3
```

In [50]:

```
cut_labels = [-1, 0, 1]
cut_bins = [-100, -0.00001, 0.00001, 100]
df_final['Final'] = pd.cut(df_final['Final'], bins=cut_bins, labels=cut_labels)
```

```
<ipython-input-50-03855659e3aa>:3: SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning
-a-view-versus-a-copy
  df_final['Final'] = pd.cut(df_final['Final'], bins=cut_bins, labels=cut_labels)
```

In [51]:

```
df_final['Final'].value_counts()
```

Out[51]:

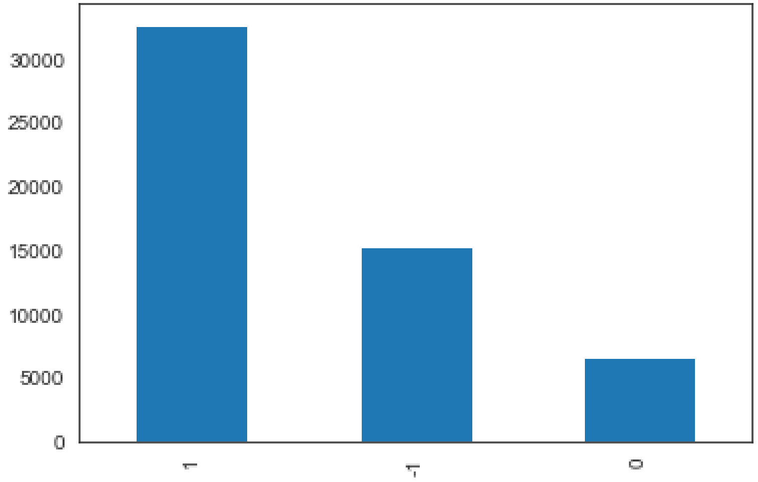
```
1     32628
-1    15267
0      6527
Name: Final, dtype: int64
```

In [52]:

```
df_final['Final'].value_counts().plot(kind='bar')
```

Out[52]:

```
<AxesSubplot:>
```



In [53]:

```
df_final.to_csv('../output/sentiment_labels.csv')
```

In [54]:

```
df_final = pd.read_csv('../output/sentiment_labels.csv', index_col=None).iloc[:,1:]
df_final.head()
```

Out[54]:

	id	text	sentiment_text	subjectivity_score	VADER	TextBlob	W2V-kNN	Final

	id	text	sentiment_text	subjectivity_score	VADER	TextBlob	W2V-kNN	Final
0	984549706549166080	Republicans and Democrats have both created ou...	republicans democrats created economic problems	0.200000	-1	1	-1	-1
1	1234653427789070336	I was thrilled to be back in the Great city of...	thrilled_back great city charlotte north_carol...	0.483333	1	1	-1	1
2	1218010753434820614	RT @CBS_Herridge: READ: Letter to surveillance...	read letter surveillance court obtained cbs ne...	0.100000	0	1	-1	0
3	1304875170860015617	The Unsolicited Mail In Ballot Scam is a major...	unsolicited mail_ballot scam major threat demo...	0.454762	-1	1	-1	-1
4	1218159531554897920	RT @MZHemingway: Very friendly telling of even...	friendly telling events comey apparent leaking...	0.425000	1	1	-1	1

In [55]:
```
df_final['VADER'].value_counts()
```

Out[55]:
```
 1    31041
-1    14684
 0     8697
Name: VADER, dtype: int64
```

In [56]:
```
df_final['TextBlob'].value_counts()
```

Out[56]:
```
 1    28536
 0    15852
-1    10034
Name: TextBlob, dtype: int64
```

In [57]:
```
df_final['W2V-kNN'].value_counts()
```

Out[57]:
```
 1    27680
-1    26742
Name: W2V-kNN, dtype: int64
```

In [58]:
```
df_final['Final'].value_counts()
```

Out[58]:
```
 1    32628
-1    15267
 0     6527
Name: Final, dtype: int64
```

In [ ]:

read data

In [1]:

```
import pandas as pd
import numpy as np
import os
import time
import numpy as np
import scipy.io
import sklearn.metrics
import sklearn
import random
import pandas as pd


# from scipy.spatial.distance import pdist
# import imbalanced_databases as imbd
import matplotlib.pyplot as plt
# import smote_variants as s
import scipy.io as scio
from PIL import Image
import pandas as pd
import numpy as np
import xlsxwriter
import scipy.io
import sklearn
import os, sys
import pickle
import random
import time
import cv2

from sklearn.pipeline import Pipeline
from sklearn.datasets import make_classification
from sklearn.metrics import (classification_report,
                             confusion_matrix,
                             recall_score,
                             accuracy_score,
                             make_scorer,
                             roc_auc_score)
```

```
from sklearn.model_selection import (train_test_split,
                                    cross_validate,
                                    GridSearchCV,
                                    RepeatedStratifiedKFold,
                                    cross_val_score,
                                    validation_curve)

# dealing with imbalanced dataset
from imblearn.under_sampling import RandomUnderSampler
from imblearn.over_sampling import SMOTE

# models
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import SGDClassifier
from sklearn.ensemble import (GradientBoostingClassifier,
                              RandomForestClassifier,
                              AdaBoostClassifier,
                             VotingClassifier)

from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import RidgeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn import tree
from sklearn.neighbors import (KNeighborsClassifier,
                               NearestCentroid,
                               NeighborhoodComponentsAnalysis)


random.seed(2021)
```

In [2]:

```
df_raw=pd.read_csv('../output/tweets_with_topic_label.csv')
df_raw=df_raw.drop(columns=['text'])
df_raw['id']=df_raw['id'].astype('int')
df_raw.head()
```

Out[2]:

	id	isRetweet	isDeleted	device	favorites	retweets	date	isFlagged	Topic
0	98454970654916608	f	f	TweetDeck	49	255	2011-08-02 18:07:48	f	0

	id	isRetweet	isDeleted	device	favorites	retweets	date	isFlagged	Topic
1	1234653427789070336	f	f	Twitter for iPhone	73748	17404	2020-03-03 01:34:50	f	6
2	1218010753434820614	t	f	Twitter for iPhone	0	7396	2020-01-17 03:22:47	f	6
3	1304875170860015617	f	f	Twitter for iPhone	80527	23502	2020-09-12 20:10:58	f	5
4	1218159531554897920	t	f	Twitter for iPhone	0	9081	2020-01-17 13:13:59	f	9

In [3]:

```
df_senti=pd.read_csv('../output/sentiment_labels.csv',index_col=None)
df_senti['id']=df_senti['id'].astype('int')
df_senti.head()
```

Out[3]:

	Unnamed: 0	id	text	sentiment_text	subjectivity_score	VADER	TextBlob	W2V-kNN	Final
0	0	98454970654916608	Republicans and Democrats have both created ou...	republicans democrats created economic problems	0.200000	-1	1	-1	-1
1	1	1234653427789070336	I was thrilled to be back in the Great city of...	thrilled_back great city charlotte north_carol...	0.483333	1	1	-1	1
2	2	1218010753434820608	RT @CBS_Herridge: READ: Letter to surveillance...	read letter surveillance court obtained cbs ne...	0.100000	0	1	-1	0
3	3	1304875170860015616	The Unsolicited Mail In Ballot Scam is a major...	unsolicited mail_ballot scam major threat demo...	0.454762	-1	1	-1	-1
4	4	1218159531554897920	RT @MZHemingway: Very friendly telling of even...	friendly telling events comey apparent leaking...	0.425000	1	1	1	1

In [4]:

```
df=df_raw.set_index('id').join(df_senti.set_index('id'),how='left')
df=df.drop(columns=['Unnamed: 0','text','sentiment_text','VADER','TextBlob','W2V-kNN'])
df.head()
```

Out[4]:

	isRetweet	isDeleted	device	favorites	retweets	date	isFlagged	Topic	subjectivity_score	Final
id										
1698308935	f	f	Twitter Web Client	939	519	2009-05-04 18:54:25	f	3	0.497222	1.0
1701461182	f	f	Twitter Web Client	259	34	2009-05-05 01:00:10	f	1	0.454545	1.0
1737479987	f	f	Twitter Web Client	37	15	2009-05-08 13:38:08	f	3	0.420000	1.0

id	isRetweet	isDeleted	device	favorites	retweets	date	isFlagged	Topic	subjectivity_score	Final
1741160716	f	f	Twitter Web Client	29	11	2009-05-08 20:40:15	f	1	0.454545	0.0
1773561338	f	f	Twitter Web Client	1877	1321	2009-05-12 14:07:28	f	2	0.000000	-1.0

In [5]:

```
df['isDeleted'].value_counts()
# imbalanced data set, consider using smote?
```

Out[5]:

```
f    55480
t     1092
Name: isDeleted, dtype: int64
```

In [6]:

```
from sklearn import preprocessing
df=df.apply(preprocessing.LabelEncoder().fit_transform)
df.head()
```

Out[6]:

id	isRetweet	isDeleted	device	favorites	retweets	date	isFlagged	Topic	subjectivity_score	Final
1698308935	0	0	13	904	519	0	0	3	864	2
1701461182	0	0	13	259	34	1	0	1	694	2
1737479987	0	0	13	37	15	2	0	3	565	2
1741160716	0	0	13	29	11	3	0	1	694	1
1773561338	0	0	13	1645	1301	4	0	2	0	0

In [7]:

```
X=df.drop(columns=['isDeleted'])
Y=df['isDeleted']
print('majority train class: %d' % np.sum(Y == 0))
print('minority train class: %d' % np.sum(Y == 1))
```

```
majority train class: 55480
minority train class: 1092
```

In [8]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test= train_test_split(X,Y,test_size=0.20,random_state=0)
```

```
X_train.shape, Y_train.shape, X_test.shape, Y_test.shape
print('majority train class: %d' % np.sum(Y_train == 0))
print('minority train class: %d' % np.sum(Y_train == 1))
print('majority test class: %d' % np.sum(Y_test == 0))
print('minority test class: %d' % np.sum(Y_test == 1))
#imbalanced dataset
```

```
majority train class: 44387
minority train class: 870
majority test class: 11093
minority test class: 222
```

In [9]:

```
from imblearn.over_sampling import SMOTE
# using SMOTE
smt = SMOTE()

# fit and apply the transform
X_train, Y_train = smt.fit_resample(X_train, Y_train)
X_test, Y_test = smt.fit_resample(X_test, Y_test)

print('majority train class: %d' % np.sum(Y_train == 0))
print('minority train class: %d' % np.sum(Y_train == 1))
print('majority test class: %d' % np.sum(Y_test == 0))
print('minority test class: %d' % np.sum(Y_test == 1))
```

```
majority train class: 44387
minority train class: 44387
majority test class: 11093
minority test class: 11093
```

In [10]:

```
y_train=Y_train
y_test=Y_test
```

models

In [11]:

```
# kNN
from sklearn.neighbors import NearestCentroid
import numpy as np
from sklearn.metrics import classification_report

clf = NearestCentroid()
start_time=time.time()
clf.fit(X_train, Y_train)
```

```
NearestCentroid()
print("Training  model takes %s seconds" % round((time.time() - start_time),3))

start = time.time()
pre=clf.predict(X_test)
end = time.time()
print("Predicting test data takes %s seconds" % round((end - start),3))
print(classification_report(Y_test,pre))
```

```
Training  model takes 0.016 seconds
Predicting test data takes 0.005 seconds
              precision    recall  f1-score   support

           0       0.81      0.60      0.69     11093
           1       0.68      0.86      0.76     11093

    accuracy                           0.73     22186
   macro avg       0.75      0.73      0.73     22186
weighted avg       0.75      0.73      0.73     22186
```

In [12]:

```
# SGD with penalty=l1
from sklearn.linear_model import SGDClassifier

clf = SGDClassifier(loss="log", penalty="l1", max_iter=200, shuffle=True, class_weight='balanced')
start_time=time.time()
clf.fit(X_train, Y_train)
print("Training  model takes %s seconds" % round((time.time() - start_time),3))

from sklearn.metrics import classification_report
start = time.time()
pre=clf.predict(X_test)
end = time.time()
print("Predicting test data takes %s seconds" % round((end - start),3))
print(classification_report(Y_test,pre))
```

```
Training  model takes 2.98 seconds
Predicting test data takes 0.003 seconds
              precision    recall  f1-score   support

           0       0.87      0.43      0.57     11093
           1       0.62      0.94      0.75     11093

    accuracy                           0.68     22186
```

```
   macro avg       0.74      0.68      0.66     22186
weighted avg       0.74      0.68      0.66     22186

/Users/aprilyang/.local/lib/python3.8/site-packages/sklearn/linear_model/_stochastic_gradient.py:570: ConvergenceWarn
ing: Maximum number of iteration reached before convergence. Consider increasing max_iter to improve the fit.
  warnings.warn("Maximum number of iteration reached before "
```

In [13]:

```
# SGD with penalty=l2
from sklearn.linear_model import SGDClassifier

clf = SGDClassifier(loss="log", penalty="l2", max_iter=200, shuffle=True, class_weight='balanced')
start_time=time.time()
clf.fit(X_train, Y_train)
print("Training  model takes %s seconds" % round((time.time() - start_time),3))

from sklearn.metrics import classification_report
start = time.time()
pre=clf.predict(X_test)
end = time.time()
print("Predicting test data takes %s seconds" % round((end - start),3))
print(classification_report(Y_test,pre))
```

```
Training  model takes 2.202 seconds
Predicting test data takes 0.003 seconds
              precision    recall  f1-score   support

           0       0.49      0.97      0.65     11093
           1       0.00      0.00      0.00     11093

    accuracy                           0.48     22186
   macro avg       0.25      0.48      0.33     22186
weighted avg       0.25      0.48      0.33     22186

/Users/aprilyang/.local/lib/python3.8/site-packages/sklearn/linear_model/_stochastic_gradient.py:570: ConvergenceWarn
ing: Maximum number of iteration reached before convergence. Consider increasing max_iter to improve the fit.
  warnings.warn("Maximum number of iteration reached before "
```

In [14]:

```
# DecisionTrees
from sklearn import tree

clf = tree.DecisionTreeClassifier(max_depth = 30, min_samples_leaf=2, max_leaf_nodes=3, class_weight='balanced')
start_time=time.time()
clf = clf.fit(X_train, Y_train)
print("Training  model takes %s seconds" % round((time.time() - start_time),3))
```

```python
from sklearn.metrics import classification_report
start = time.time()
pre=clf.predict(X_test)
end = time.time()
print("Predicting test data takes %s seconds" % round((end - start),3))
print(classification_report(Y_test,pre))
```

```
Training  model takes 0.159 seconds
Predicting test data takes 0.003 seconds
              precision    recall  f1-score   support

           0       0.83      0.81      0.82     11093
           1       0.82      0.83      0.83     11093

    accuracy                           0.82     22186
   macro avg       0.82      0.82      0.82     22186
weighted avg       0.82      0.82      0.82     22186
```

In [15]:

```python
# RandomForestClassifier
from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier(n_estimators=300, class_weight='balanced')
start_time=time.time()
clf = clf.fit(X_train, Y_train)
print("Training  model takes %s seconds" % round((time.time() - start_time),3))

from sklearn.metrics import classification_report
start = time.time()
pre=clf.predict(X_test)
end = time.time()
print("Predicting test data takes %s seconds" % round((end - start),3))
print(classification_report(Y_test,pre))
```

```
Training  model takes 39.3 seconds
Predicting test data takes 0.867 seconds
              precision    recall  f1-score   support

           0       0.72      0.98      0.83     11093
           1       0.97      0.62      0.76     11093

    accuracy                           0.80     22186
   macro avg       0.85      0.80      0.80     22186
```

```
weighted avg       0.85      0.80      0.80     22186
```

In [16]:

```
# AdaBoostClassifier
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import AdaBoostClassifier

clf = AdaBoostClassifier(n_estimators=500)
start_time=time.time()
clf = clf.fit(X_train, Y_train)
print("Training  model takes %s seconds" % round((time.time() - start_time),3))

from sklearn.metrics import classification_report
start=time.time()
pre=clf.predict(X_test)
end = time.time()
print("Predicting test data takes %s seconds" % round((end - start),3))
print(classification_report(Y_test,pre))
```

```
Training  model takes 32.648 seconds
Predicting test data takes 1.098 seconds
              precision    recall  f1-score   support

           0       0.89      0.85      0.87     11093
           1       0.86      0.89      0.87     11093

    accuracy                           0.87     22186
   macro avg       0.87      0.87      0.87     22186
weighted avg       0.87      0.87      0.87     22186
```

In [17]:

```
# GBM

# define the model
def train_model_gb(X, y):
    model_gb = GradientBoostingClassifier(n_estimators=500)
    model_gb.fit(X, y)

    return model_gb
# train
training_gbm = time.time()
model_gb = train_model_gb(X_train, Y_train)
```

```
print("Train the Gradient Boosting Model takes %s seconds" % round((time.time() - training_gbm), 3))

y_pred = model_gb.predict(X_test)

# print the confusion matrix
print(confusion_matrix(y_true=Y_test, y_pred=y_pred))
print(classification_report(y_true=Y_test, y_pred=y_pred))
```

```
Train the Gradient Boosting Model takes 63.676 seconds
[[10026  1067]
 [ 1292  9801]]
              precision    recall  f1-score   support

           0       0.89      0.90      0.89     11093
           1       0.90      0.88      0.89     11093

    accuracy                           0.89     22186
   macro avg       0.89      0.89      0.89     22186
weighted avg       0.89      0.89      0.89     22186
```

In [16]:

```
weighted_svm_best = SVC(
gamma = 'scale',
class_weight = {
    0: 1092.0,
    1: 55480.0
},
probability=True
)

start_time = time.time()

# fit svm model
weighted_svm_best.fit(X_train, y_train)

print("Training  model takes %s seconds" % round((time.time() - start_time),3))
print('Testing Accuracy of weighted SVM on test set: {:.3f}'
    .format(weighted_svm_best.score(X_test,y_test)))

start = time.time()

# make prediction
weighted_svm_pred = weighted_svm_best.predict(X_test)
```

```
end = time.time()

weighted_svm_predprob = weighted_svm_best.predict_proba(X_test)[:,1]

print("Predicting test data takes %s seconds" % round((end - start),3))
print('Classification error rate:', np.mean(np.array(y_test) != weighted_svm_pred))
print('Classification report \n', classification_report(y_test, weighted_svm_pred))

print('Confusion Matrix \n', confusion_matrix(y_test, weighted_svm_pred))
print('AUC is: {:.4f}'.format(roc_auc_score(y_test, weighted_svm_predprob)))

# callModel(eclf2, 'eclf2', X_train, X_test, y_train, y_test)
```

In [ ]: