****

DSC550 - Data Mining Coursework Project :

Trump Tweet Deets... 2014-2021

****

==========

Requirement...

==========

You have made it to the final weeks of the course and the time has come to submit your final project milestone! The only new requirement for the final milestone is to include a conclusion/summary. At a minimum make sure that you answer the following questions:

- What does the analysis/model building tell you?
- What are your recommendations?
- How would you pitch this business problem to a group of stakeholders to gain buy-in to proceed?
- Why should someone in the business care about this solution?
- What are some of the potential challenges or additional opportunities that need to be explored?

You will need to submit the following as part of your final submission for Milestone 5:

- Case Study Report
- Introduction
- Organized and Detailed Summary of Milestones 1- 3
- Business Problem/Data
- Graphical Analysis
- Dimensionality & Feature Reduction and Feature Engineering
- Model Selection & Evaluation
- Conclusion (Findings, Summarization)
- Code – either via PDF or a Notebook

Any additional supporting documents Your case study report should be a minimum of 3-5 pages, along with visuals created to support your analysis. Submit Milestone 5 to the submission link.

==========

Motivations...

==========

Source: May 04, 2021 Donald Trump's 'social media platform' has launched and it's just a blog

After months of promising his own social media network for banned posters, former President Donald Trump on Tuesday launched a new section of his website — essentially a standard-issue blog.

The "platform" is styled like of Twitter but hosted as a running blog of Twitter-length commentary from Trump. People can sign up for post alerts on the platform through their email and phone numbers and are allegedly able to like them, although that function doesn't appear to work as of publication. Users are also allowed to share Trump's posts on Facebook and Twitter. The Twitter sharing option doesn't currently work, but Facebook's does allow people to share Trump's posts.

"Generally, sharing content from the website reference is permitted as long as the material does not otherwise the Twitter Rules," a Twitter spokesperson told *The Verge* on Tuesday. Facebook did not immediately respond to a request for comment.

THE NEW "PLATFORM" IS EFFECTIVELY A BLOG STYLED LIKE A GENERIC VERSION OF TWITTER

Even though the platform formally launched on Tuesday, there are posts dating back to March 24th. Trump's latest post is a video advertising his new platform, calling it "a place to speak freely and safely, straight from the desk of Donald J. Trump." According to *Fox News*, Trump will "eventually" be able to communicate with his supporters directly, although it's not clear how that will happen. Trump's press office did not immediately respond to a request for comment

==========

Problem Definition...

==========

Now in 2021, Trump stands banned from some social media platforms (Twitter, Facebook, et al) – however his usage of Twitter as a platform was unprecedented, more so as a leader of free world. He used tweets to praise, cajole, entertain, lobby, curse – he used twitter to amplify his scorn – which followers might refer to as "speak his mind". Trump's Twitter theatricals led to this rich repo of "insults" targeted at a variety of stakeholders.

Even after the end of presidency, Trump theatricals though has lost a bit of steam being off Twitter, yet this hasn't necessarily translated to loss of "popularity". Trump, even today, does have huge enviable supporter base, although he is barred from Twitter and Facebook. To me, this likens to a "Politician Rockstar" -- who has a fan-base... his cult definitely has the potential of getting emulated and opportunity to encash (if the params/ variables are designed/ decoded well enough)

I see Trump's tweeting behavior as a business opportunity, which any politician may try to replicate in future. I found the tweeting phenomenon "interesting", since the source was "the most powerful (political) person on earth", and I was overdosed with data mining concepts ;-) ... so this was destined to be an opportunity to merge the two events.

The scenario I imagine is sometime in future (say 2040):

- Trump's tweeting, theatricals, insults, etc are a model for politicians to replicate -- but minus the human-brain space/ effort

- What if a machine/ a code can perform the same theatricals for the next US Presidential candidate i.e. generate similar levels of engagement aka divisiveness, hype, confusion, whims, fancies, bursts -- create the *"next Trump"*
- Business Opportunity: Can I develop a predictor / recommendation engine that ranges amongst the subject areas of Trump's targets. If I can somehow identify the subjects or personas or patterns that Trump targeted, I can probably extrapolate it to current timeline to carve out a `tweet bot strategy` for the presidential campaign and maybe tenure too

Well, from an academic exercise aspect, recommendations/ predictions in reality can be applied to any area -- say stocks, exchange indices, commodity prices, currency fluctuations, international trade volumes, etc. My point is I am assuming that these Tweets can impact relations and affect real business amongst stakeholders, thus has a $$ value associated, which I am trying to encash through this academic pilot project.

==========

Dataset...

==========

As a political figure, Donald J. Trump used Twitter to praise, to cajole, to entertain, to lobby, to establish his version of events — and, perhaps most notably, to amplify his scorn. This list documents the verbal attacks Mr. Trump posted on Twitter, from when he declared his candidacy in June 2015 to Jan. 8, when Twitter permanently barred him.

Based on an analysis of tweets since Mr. Trump declared his candidacy for president, on June 16, 2015. Retweets are not included. Some names may be omitted. This was first published in 2016; some of the people insulted have since died, and some people's titles or public roles have changed. Source: Trump Twitter Archive

References:

1. https://www.nytimes.com/interactive/2021/01/19/upshot/trump-complete-insult-list.html
2. https://www.kaggle.com/ayushggarg/all-trumps-twitter-insults-20152021

==========

Analysis Outline / Objectives...

==========

1. Dataset Exploration and Cleaning
    - Who - targets?
    - How - frequent?
    - Most used words?
    - Most frequent targets?
    - What changes is you remove grammar from vocab? (prepositions are junk ! -- is this true?)
2. LDA & Sentiment analysis
    - Sentiment Analysis
    - Bag-of-Words to serve as a dictionary
    - does token-words change after cleansing?
3. Model selection/ evaluation and feature engineering
4. Demo of tweet prediction

Refer "Workings Detail" *below* for details of outline of the implementation...

[A] Let's read/ process

 # 1.a - Start with reading/ cleaning / processing the csv file...

[A.1] Data Cleaning

 # 1.b - If we cleanse raw data i.e. rationalize the words, does any of our prior data exploration change/ deviate significantly?

Observations -

- 1.a Reading the csv is fairly straightforward, I did have to ignore/suppress (library/package related) warnings
- 1.b Once my data does not contain any unwanted characters, I will perform Lemmatization to reduce inflected words to their root words. With data cleaning, formatted - cleansed outputs seem to nullify the grammatical aspects of the tweet, and the token-words are just the dictionary of words (*like a bag-of-words?*)
- 1.b `top-insults` what does the vocabulary of world's most-powerful-man consist of?
- 1.b `about` is the most used word in Trump tweets, with `absurd` following behind 2nd -- quite behind when compared across the entire tweeting window 2014-2021

[B] Okay, now let's try to see if the dataset has any patterns

 # 2 - patterns/trends with tweets -- *can the dataset answer following questions*

- does the count / frequency of tweets change/ spike or remain consistent over the entire period?
- who are the targets? how does the target trend change over the years? Are there any repeat / persistent targets in the tweets? does he really target or is it hoot-and-scoot for getting attention from his supporters? *(EDA - visualize Trump tweets/ targets)*

Observations -

- except for 2017, count of tweets has increased for Trump during the tenure (P.S. Trump did delete/ modify a lot of his tweets, this dataset doesnt account for deleted tweets, which were probably a recurring behavior for Trump mid-presidency?)

- `most-insulted Trump targets : media , democrats , hillary , russia , joe-biden` were primary trump targets across the 2014-2021 tweeting window

- Tweet count `Insults-per-target` : Trump wasn't a hoot-and-scoot type tweeter, he was persistent for some targets, and mostly he was deliberate...
- *thinking more of this - why do I say across 2014-2021, what if the scope was reduced to each year between 2014-2021 period* will the targets change across years? is it only Trump or does he have additional vocabulary due to an advisor? since the tweet volume is changing rather increasing... is Trump really doing all this ? or did he deploy a bot/program to do the 'Trump-thing' hereon (*wow! so there's a market for this kind of bot-behavior...*)

[C.1.1] Exploratory Analysis

 # 3.a - Sentiment analysis | token-words

Observations -

As we can see, more positive tweets have been tweeted since 2017. There's certaintly a peak of negativity from 2014 to 2015 which has been mantained untill 2017 but then the negativity decreases a lot and very abruptly. But as we can see in the next cell, the number of tweets per year have been increasing since 2014. Of course, there are not much 2021 tweets given elections and Twitter ban, but in any case there's a tendency of lowering the negativity.

[C.1.2] Sentiment Analysis - VADER

 # 3.b - "Trump faces Vader !" *vaderSentiment analysis* -- VADER *Valence Aware Dictionary and Sentiment Reasoner*

VADER (Valence Aware Dictionary and Sentiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media. It is used for sentiment analysis of text which has both the polarities i.e. positive/negative. VADER is used to quantify how much of positive or negative emotion the text has and also the intensity of emotion. Advantages Here are the advantages of using VADER which makes a lot of things easier: It does not require any training data. It can very well understand the sentiment of a text containing emoticons, slangs, conjunctions, capital words, punctuations and much more. It works excellent on social media text. VADER can work with multiple domains.

Observations -

- *pos*: The probability of the sentiment to be positive.
- *neu*: The probability of the sentiment to be neutral.
- *neg*: The probability of the sentiment to be negative.
- *compound*: The compound score is computed by summing the valence scores of each word in the lexicon, adjusted according to the rules, and then normalized to be between -1 (most extreme negative) and +1 (most extreme positive).

Notice that the *pos*, *neu* and *neg* probabilities add up to 1. Also, the *compound* score is a very useful metric in case we want a single measure of sentiment. Typical threshold values are the following:

- *positive*: compound score>=0.05

- *neutral*: compound score between -0.05 and 0.05
- *negative*: compound score<=-0.05

Now with this results we'll see what's Trump's evolution by plotting the compound sentiment score by year.

[C.1.3] Sentiment Analysis - WordCloud

 # 3.c - WordCloud *word visuals*

Observations -

- The generated images are somewhat aligned with EDA analysis done earlier i.e. `media` , `democrats` , `hillary` , `russia` , `joe-biden` in descending order are the `most-insulted targets` . The size of these *names* on the WordCloud image indicates the relevance / frequency of target. This is very much similar to observations made with the dataset exploratory trends.
- If we see the words - size and intensity `fake` , `new` , `witch` , `hunt` , `crooked` , `people` are some of the most used words across the tweets
- Wordcloud results are aligned with Trump's image I witnessed on news channels, analyst coverages and other media portals including social media platforms

[C.2] Latent Dirichlet Allocation (LDA) Modeling

What is the core of Trump insults? Can these tweets be programmatically organized into topics? Can patterns be discovered that may be useful for further analysis/modeling?

Observations -

Categorising words into topics, we had discreet tweets which when modelled into topics are indicated through distinct bubbles on left which are distinguished with varied intertopical distance amongst other bubbles.

[D] Model Selection and Evaluation

Why choose Random Forest ?

The term “Random Forest Classifier” refers to the classification algorithm made up of several decision trees. The algorithm uses randomness to build each individual tree to promote uncorrelated forests, which then uses the forest's predictive powers to make accurate decisions. In this instance, although source of tweets is Trump, the targets are assumed to be uncorrelated given reality of Trump whims.

For model evaluation, I did get reasonable high accuracy and precision scores. I don't think model development applies (or I couldn't make much headway in this aspect of the dataset usecase/activity). My initial goal was to use the tweet topics as features with the target as the impact variable. So my presumed theory I was aiming for is with a bag-of-words and having known historical targets, the model would predict who would be the next target. Been thinking through, i realized my folly and pivoted to drop the feature selection/ engineering aspect of model. Instead I pivoted to tweet generating/ predicting using FastAI. So the bag-of-words tries to image a persona,which is the desired behavior of a tweet bot.

[E] DEMO : Tweet Generator using FastAI

As a selling proposition for this product This BOT-like product attempts to offload the "thinking" (aka targeting/ insulting?) abilities to the bot along with features like sentiment (bag-of-words), and choice of target. Well, the choice of target is subject to many external factors in reality, but this is just an academic scope, so a theoretical exercise aiming to mimic Donald Trump's theatrics ;)

Business Plan / Pitch -

- I have assumed this dataset `Trump Tweet 2014-2021` to be an indicator of Trump's 'real' persona -- which is desired by a future USA president. A president-in-future *(say 2050?)* wants to replicate Trump's theatricals in spirit, but doesn't want to actually use his own brains, rather led a machine trained model do it for him -- its a **`Trump T-BOT`**
- `Trump T-BOT` is trained on variety of Trump tweets or can use a variety of "mood" or "sentiment" or "tone" as desired by its user. I have used a language based package called FastAI (text based classification model) for tweet generation. I will be training this model with different versions of Tweet to observe the verbiage of tweets and its tone changes
- How is this relevant? What is the use of all this?
  - To imagine a situation say 2050, the president of US at this time wants to "use" a bot to create somewhat similar social media theatricals. I've assumed that Trump's tweet activity as a desired/ model behavior for future US presidents, which can be offloaded to a bot that is pre-programmed per a desired prior persona.
  - Why only Trump tweets? We can actually pick any such character -- who people would desire to replicate. And to build Tweet-BOT product which specializes in pick-a-persona and then delivers *'cruise-mode'* for twitter/ social media bursts
  - Prior analyses help in identifying the tone, assigning the right sentiment / model score that decides on the controlling limits and programmatic controls with the model
  - Valence Aware Dictionary and Sentiment Reasoner (VADER) Sentiment Analysis for trend of positive, negative sentiment over years. Random Forest and generative statistical modeling -- Latent Dirichlet Allocation (LDA) analysis. These analysis, observations are done for the scope of academic exercise. However at scale, and with due diligence to inputs and control params can form the requirement skeleton for a social media bot which intends to replicate Trump's Twitter theatricals

- Demonstration of a tweet generator (using FastAI). Compare / highlight the differences in output/ tone based on the training dataset.
  - *controlling the training dataset* -- in order to meet desired notes/ sentiments in the generated tweet

Observations -

When the model is trained on the dataset of Trump's own tweets, the generated tweet seem to match Trump's public image. With the different dataset, the tweets didn't exactly align or doesn't seem to be perceived the same way as intended. The difference in either datasets apart from the volume of rows, is the source of truth. Media tweets of trump are part of the dataset where the generated tweets seem "dreamy" unlike Trump

- I extracted a random twitter dataset from variety of sources.. It includes a healthy mix of his tweets as well as tweets/ snippets about him (primarily from media outlets). My idea of this exercise is to demonstrate the impact on the tone/ lexicon of the tweet when the model is trained on this *mixed* tweet dataset vis-a-vis the tweet-insult dataset

- Training the FastAI model on the datasets is a tedious task. For a ~40k tweet dataset, running 5 iterations took over 2.5 hours with a Macbook Pro machine, indicating the compute process intensity/latency of the predicting model

- The model seems very disconnected to 'real' Trump when trained with *mixed* tweet dataset. The T-BOT spit out : `you are LAT.MS correctly , i think` ; while when the model is trained on tweet-insult dataset it spits out : `know what to do with them` -- quite a proximation of the real verbosity

==========

Conclusion...

==========

Imagine a world, where a tweet can impact markets, indices and stocks; affect trade relations, commodity prices -- a tweet that can make or mar business - real business dollars. What if this tweet can be controlled so that the impact too is patterned/ predictable. So, assume a future politician needing to replicate Trump theatricals, but instead of actually using own mind-space or brainpower, the person intends to try/ trust a machine to do this 'mindful' significant job -- it means real business dollars.

This imagination/ motivation for this task was driven by Trump's own social media platform launch in May 2021 - which seems like a repository of his tweets/anecdotes, and glaringly lacking the engagement aspect associated with social media platforms. *Its ironic that as of today, Trump's platform has shutdown.*

I started off the task with aim to analyze the dataset, find variables to control, try to programmatize my approach for replicability, accuracy and explainability aspects. LDA modeling, sentiment analysis ... bag-of-words are aimed at explaining the dataset - key targets, insult keywords, most-insulted targets. I started off random forest modeling trying to identify key features in the model, which I can use for affecting the impact variable. Feature engineering -- this is one aspect of the exercise which I could have dug deeper with more time, this would have helped me in identifying control parameters for the model. I did attempt to demo *a very basic* usecase of how a controlled dataset predicts / directionally mimics the character persona. My intention with the FastAI snippet to generate tweets, was to get a stakeholder buy-in with the end state of the model. The next step to action it would be a scheduler which generates text at a politician/ aspirant controlled cadence.

Given academic scope of the task, each of the analyses scopes has ton of opportunity of being iterated and scoped to elaborate plans. However I limited it to the course tenure and the milestone objectives.

****

`<<<   refer below section for detailed code workings/ analyses   >>>`

****

==========

Workings Detail...

==========

==========

#1a. -- Workings Detail... Ref Section Heading: [A] Let's read/ process...

==========

In [1]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
#import relevant libraries
#for text processing
import string
import re
import nltk
from textblob import TextBlob
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer

import numpy as np # linear algebra
import pandas as pd # data processing
import seaborn as sns; sns.set(style="ticks", color_codes=True)
import matplotlib.pyplot as plt
import plotly.graph_objs as go
from plotly.subplots import make_subplots
import seaborn as sns
import cufflinks as cf
import plotly.express as px
%matplotlib inline

from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
init_notebook_mode(connected = True)
cf.go_offline()

from nltk.corpus import stopwords
from nltk.util import ngrams
from nltk.stem import SnowballStemmer
from collections import defaultdict
from wordcloud import WordCloud

#for feature selection
from sklearn import decomposition

#for model development
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import classification_report
#%matplotlib inline
```

In [3]:
```
#read the file and basic explore...
df=pd.read_csv("trump_insult_tweets_2014_to_2021.csv")
```

In [4]:
```
df.head()
```

Out[4]:

	Unnamed: 0	date	target	insult	tweet
0	1	2014-10-09	thomas-frieden	fool	Can you believe this fool, Dr. Thomas Frieden ...
1	2	2014-10-09	thomas-frieden	DOPE	Can you believe this fool, Dr. Thomas Frieden ...
2	3	2015-06-16	politicians	all talk and no action	Big time in U.S. today - MAKE AMERICA GREAT AG...
3	4	2015-06-24	ben-cardin	It's politicians like Cardin that have destroy...	Politician @SenatorCardin didn't like that I s...
4	5	2015-06-24	neil-young	total hypocrite	For the nonbeliever, here is a photo of @Neily...

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
Unnamed: 0    0
date          0
target        2
insult        0
tweet         0
dtype: int64
```

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10360 entries, 0 to 10359
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   Unnamed: 0  10360 non-null  int64 
 1   date        10360 non-null  object
 2   target      10358 non-null  object
 3   insult      10360 non-null  object
 4   tweet       10360 non-null  object
dtypes: int64(1), object(4)
memory usage: 404.8+ KB
```

In [7]:
```
df.shape
```

Out[7]:
```
(10360, 5)
```

In [8]:
```
df=df.dropna(axis=0)
```

In [9]:
```
from datetime import datetime
df['date'] = pd.to_datetime(df['date'], infer_datetime_format = True)
df['year']=df['date'].dt.year
df['month']=df['date'].dt.month
df['week']=df['date'].dt.isocalendar().week
df.head()
```

Out[9]:

	Unnamed: 0	date	target	insult	tweet	year	month	week
0	1	2014-10-09	thomas-frieden	fool	Can you believe this fool, Dr. Thomas Frieden ...	2014	10	41
1	2	2014-10-09	thomas-frieden	DOPE	Can you believe this fool, Dr. Thomas Frieden ...	2014	10	41
2	3	2015-06-16	politicians	all talk and no action	Big time in U.S. today - MAKE AMERICA GREAT AG...	2015	6	25
3	4	2015-06-24	ben-cardin	It's politicians like Cardin that have destroy...	Politician @SenatorCardin didn't like that I s...	2015	6	26
4	5	2015-06-24	neil-young	total hypocrite	For the nonbeliever, here is a photo of @Neily...	2015	6	26

==========

#1b. -- Workings Detail... Ref Section Heading: [A.1] Data cleaning...

==========

Let's cleanup the dataset...

In [11]:
```
#lowercase the column names
df.columns = df.columns.str.lower()
df.columns
```

Out[11]:
```
Index(['unnamed: 0', 'date', 'target', 'insult', 'tweet', 'year', 'month',
```

```
       'week'],
      dtype='object')
```

In [12]:
```
df1=df[['date', 'target', 'insult', 'tweet', 'year']]
df['tweet'] = df['tweet'].astype('str')
df['insult'] = df['insult'].astype('str')
```

In [13]:
```
# apply same function on tweet ~ insult
def clean_data(text):
    text = text.lower()  # convert all the text into lowercase
    text = text.strip()  #remove starting and trailing whitespaces
    special_char_reg = '([a-zA-Z0-9]+)' + '[!"#$%&\'()*+,-./:;<=>?@\\^_`{|}~]' + '([a-zA-Z0-9]+)'
    text = re.sub(special_char_reg, ' ', text)
    text = re.sub(r'\s+', ' ', text) #remove all line formattings
    text = re.sub(r'\d+', '', text) #remove digits
    text = re.sub(r'\w*\d\w*', '', text)
    text = re.sub(r"\w+…|…", "", text)  # remove ellipsis (and last word)
    text = re.sub(f"[{re.escape(string.punctuation)}]", "", text)
    text = ''.join(c for c in text if c not in string.punctuation)   #remove special symbols from job titles
    return text
```

In [14]:
```
# peek at "raw" tweets
tweet_df = df.tweet.apply(lambda x : clean_data(x))
tweet_df.head()
```

Out[14]:
```
0    can you believe this fool dr thomas frieden of...
1    can you believe this fool dr thomas frieden of...
2    big time in  today  make america great again p...
3    politician senatorcardin like that i said balt...
4    for the nonbeliever here is a photo of neilyou...
Name: tweet, dtype: object
```

In [15]:
```
# peek at "raw" insults
insult_df = df.insult.apply(lambda x : clean_data(x))
insult_df.head()
```

Out[15]:
```
0                                              fool
1                                              dope
2                                all talk and no action
3     politicians like cardin that have destroyed b...
4                                       total hypocrite
Name: insult, dtype: object
```

LookupError: ********** Resource punkt not found. Please use the NLTK Downloader to obtain the resource: >>> import nltk >>> nltk.download('punkt') LookupError: ********** Resource wordnet not found. Please use the NLTK Downloader to obtain the resource: >>> import nltk >>> nltk.download('wordnet')

In [16]:
```
import nltk
nltk.download('punkt')
```

Out[16]:
```
[nltk_data] Downloading package punkt to
[nltk_data]     /Users/binay.jena/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
True
```

In [17]:
```
import nltk
nltk.download('wordnet')
```

Out[17]:
```
[nltk_data] Downloading package wordnet to
[nltk_data]     /Users/binay.jena/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
True
```

In [18]:
```
from nltk import WordNetLemmatizer
def lemma(text):
    word_list = nltk.word_tokenize(text) #tokenize before lemmatization
    lemma_output = ' '.join(WordNetLemmatizer().lemmatize(word) for word in word_list)
    return lemma_output
tweet_df_lemma = tweet_df.apply(lambda x : lemma(x))
insult_df_lemma = insult_df.apply(lambda x : lemma(x))
```

In [19]:
```
from nltk import word_tokenize
def remove_stopwords_and_tokenize(text):
    the_stopwords = set(stopwords.words("english"))
    tokens = word_tokenize(text)  # tokenize
    tokens = [t for t in tokens if not t in the_stopwords]
    tokens = [t for t in tokens if len(t) > 1]
    return tokens
```

In [20]:
```
# peek at "clean" tweets
tweet_df_token = tweet_df_lemma.apply(lambda x : remove_stopwords_and_tokenize(x))
tweet_df_token.head()
```

Out[20]:
```
0    [believe, fool, dr, thomas, frieden, cdc, stat...
1    [believe, fool, dr, thomas, frieden, cdc, stat...
2    [big, time, today, make, america, great, polit...
3    [politician, senatorcardin, like, said, baltim...
4    [nonbeliever, photo, neilyoung, office, reques...
Name: tweet, dtype: object
```

In [21]:
```
# peek at "raw" insults
insult_df_token = insult_df_lemma.apply(lambda x : remove_stopwords_and_tokenize(x))
insult_df_token.head()
```

Out[21]:
```
0                                        [fool]
1                                        [dope]
2                                [talk, action]
3    [politician, like, cardin, destroyed, baltimore]
4                              [total, hypocrite]
Name: insult, dtype: object
```

Observation ^ --

- if we compare "raw" and "clean" -- tweets and insults -- (just visual inspection),
- we see the pruning has happened with grammar aspects
- the words indicating the target, the sentiment are part of the acceptable tweet and insult tokens

In [39]:
```
#Tokenization using count vectorizer
count_vect = CountVectorizer(ngram_range=(1,1))
token = count_vect.fit_transform(tweet_df_lemma)
token
```

Out[39]:
```
<10358x8977 sparse matrix of type '<class 'numpy.int64'>'
        with 301429 stored elements in Compressed Sparse Row format>
```

In [40]:
```
print('Total number of tokens/words in all the tweets - ', len(count_vect.get_feature_names()))
```

```
Total number of tokens/words in all the tweets -  8977
```

Top Insults

Fine... so many insults... its Trump.. what else should we expect from "the entertainer", except that he is kind-of-the most-powerful authority on the planet, plus all the whims and theatricals - a really tempting prime time episode, but its the US presidency with the whole world watching agape

In [41]:
```
# identify "Top Insults" i.e. impactful insults? Frequently worded insult? Target scoring?
# bag of words/ LDA for the tweets
```

In [28]:
```
#Tokenization using count vectorizer
count_vect = CountVectorizer(ngram_range=(1,1))
token = count_vect.fit_transform(insult_df_lemma)
token
```

Out[28]:
```
<10358x3827 sparse matrix of type '<class 'numpy.int64'>'
        with 44458 stored elements in Compressed Sparse Row format>
```

In [29]:
```
temp_df =  pd.DataFrame(token.toarray(), columns=count_vect.get_feature_names())
temp_df.tail()
```

Out[29]:

	abandoned	abc	abe	ability	able	abortion	about	absentee	absolute	absolutely	...	young	your	yours	yourself	youth	yovanovitch	zealand	zero	zilch	zuker
10353	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
10354	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
10355	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
10356	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
10357	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 3827 columns

In [30]:
```
#count the accurence of each token in entire corpus
count_df_insult = temp_df.apply(lambda x : x.sum())
```

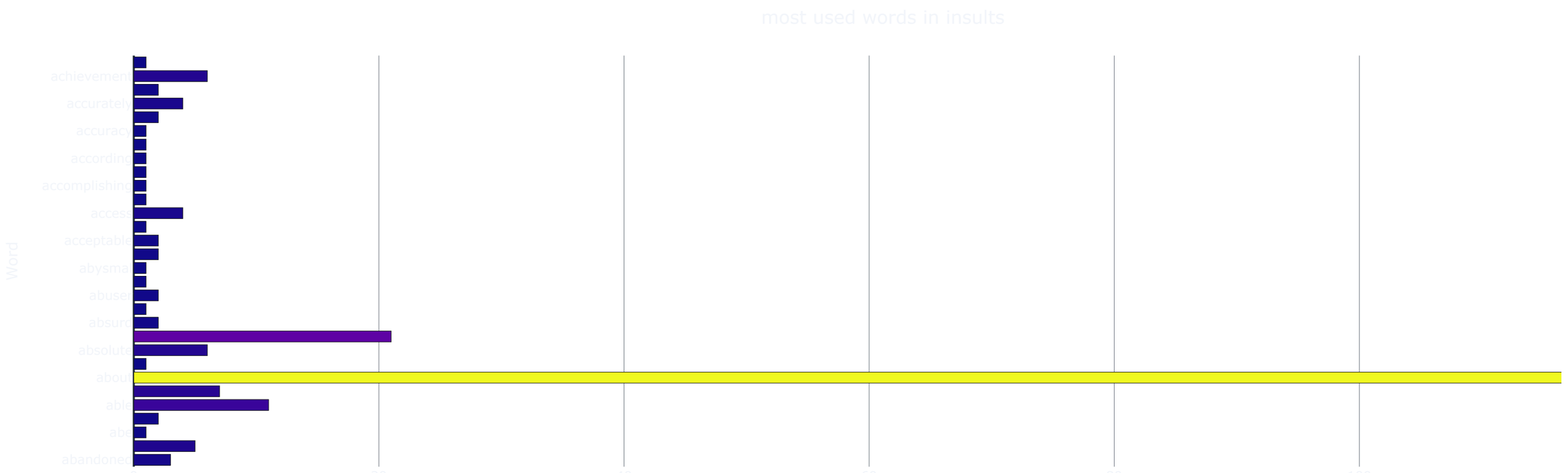
In [31]:
```
# word-counts
count_df_insult = pd.DataFrame(count_df_insult).reset_index()
count_df_insult.columns = ['Word', 'Count']
most_insults= count_df_insult.sort_values(by= 'Count', ascending=False)
most_insults.head(10)
```

Out[31]:

	Word	Count
3381	the	1746
1176	fake	1012
117	and	942
3433	to	917
2250	news	842
2314	of	834
1642	in	603
1293	for	438
1750	is	411
739	crooked	411

In [32]:

```
fig = px.bar(count_df_insult.head(30),y='Word',x='Count',
             orientation='h',color='Count')
fig.update_layout(title_text='most used words in insults',title_x=0.5,
                  template='plotly_dark')
fig.show()
```



==========

#2 - Overall trend summaries... Ref Section Heading: [B] Okay, now let's try to see if the dataset has any patterns...

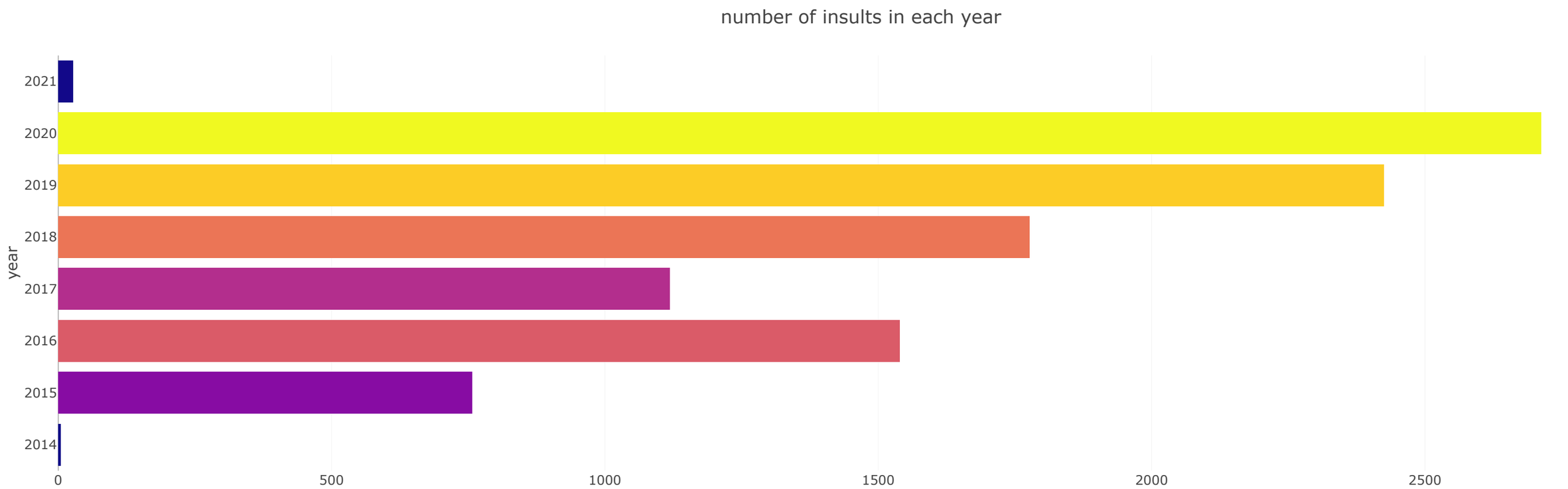
==========

In [59]:
```
df.date.dt.year.value_counts()
```

Out[59]:
```
2020    2712
2019    2425
2018    1777
2016    1539
2017    1119
2015     757
2021      27
2014       2
Name: date, dtype: int64
```

In [10]:
```
# let's see over the years 2014-2021 -- how was Trump ~ Twitter Insult overall trends

import plotly.express as px
df_year = pd.DataFrame(df.groupby('year')['insult'].count().reset_index())
df_year.columns=['year','count']
df_year = df_year.sort_values(['count'],ascending=True)
```

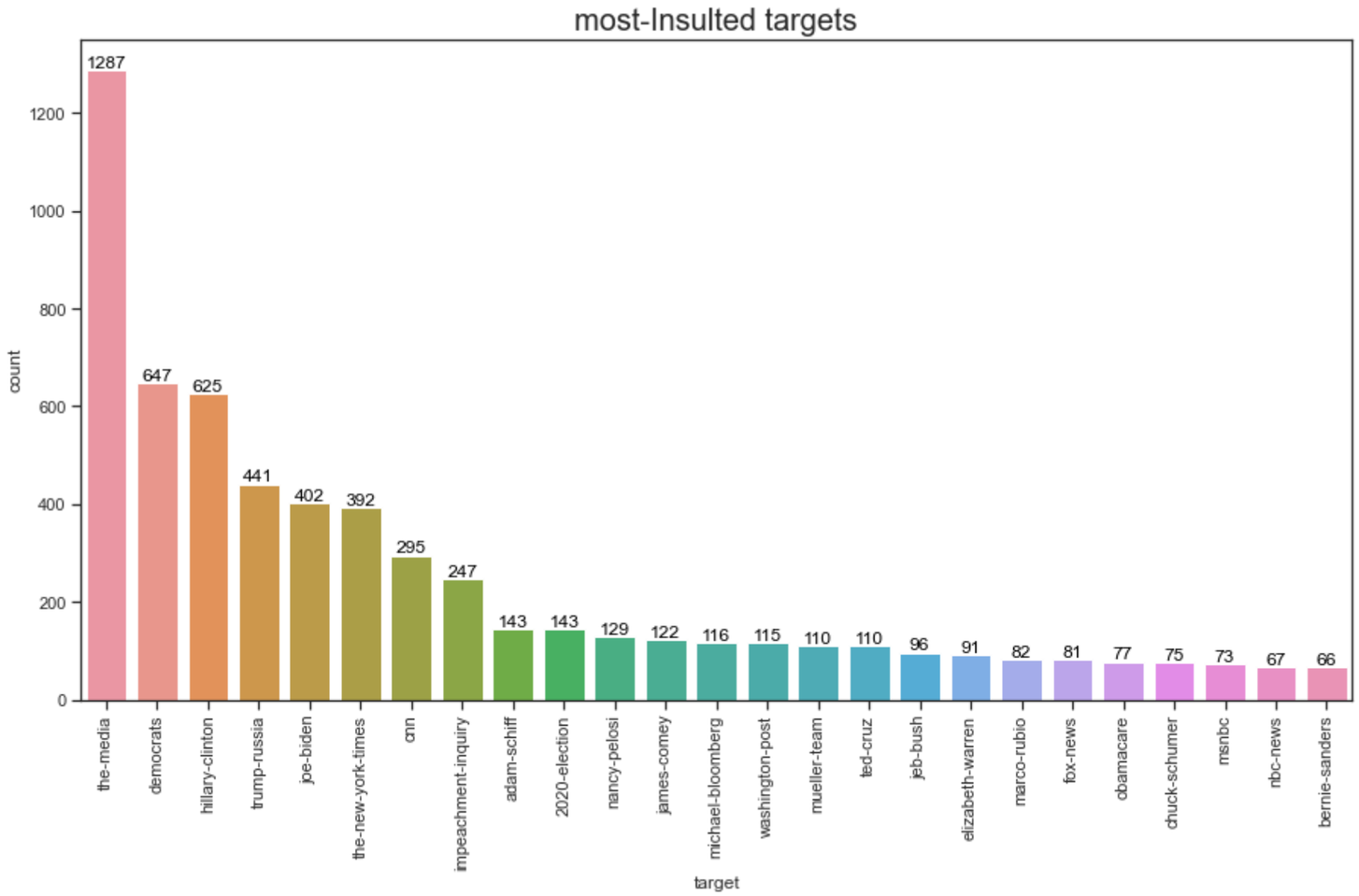
```
fig = px.bar(df_year,y='year',x='count',
             orientation='h',color='count')
fig.update_layout(title_text='number of insults in each year',title_x=0.5,
                  template='none')
fig.show()
```



Most-Insulted targets by Trump

In [17]:
```
# there were instances when tweets were deleted in reality,
# unable to verify if the dataset contains such deleted tweet instances as well
# 2020 understandable because of campaign year, but the trend does seem to be increasing over the years of tenure
```

In [18]:
```
# let's see insulted targets counted across the years in the dataset 2014-2021
most_Insulted = df.groupby(['target'])['Unnamed: 0'].count().reset_index(name='count').sort_values(by='count',ascending=False)

fig_dims = (15, 8)
fig, ax1 = plt.subplots(figsize=fig_dims)
figure1= sns.barplot(x='target',y='count' , ax=ax1 , data=most_Insulted.head(25))
for item in figure1.get_xticklabels():
    item.set_rotation(90)

ax = plt.gca()

# Iterate through the list of axes' patches
for p in ax.patches:
    ax.text(p.get_x() + p.get_width()/2., p.get_height(), '%d' % int(p.get_height()),
            fontsize=12, color='black', ha='center', va='bottom')

plt.title('most-Insulted targets',size='20')

plt.show()
```

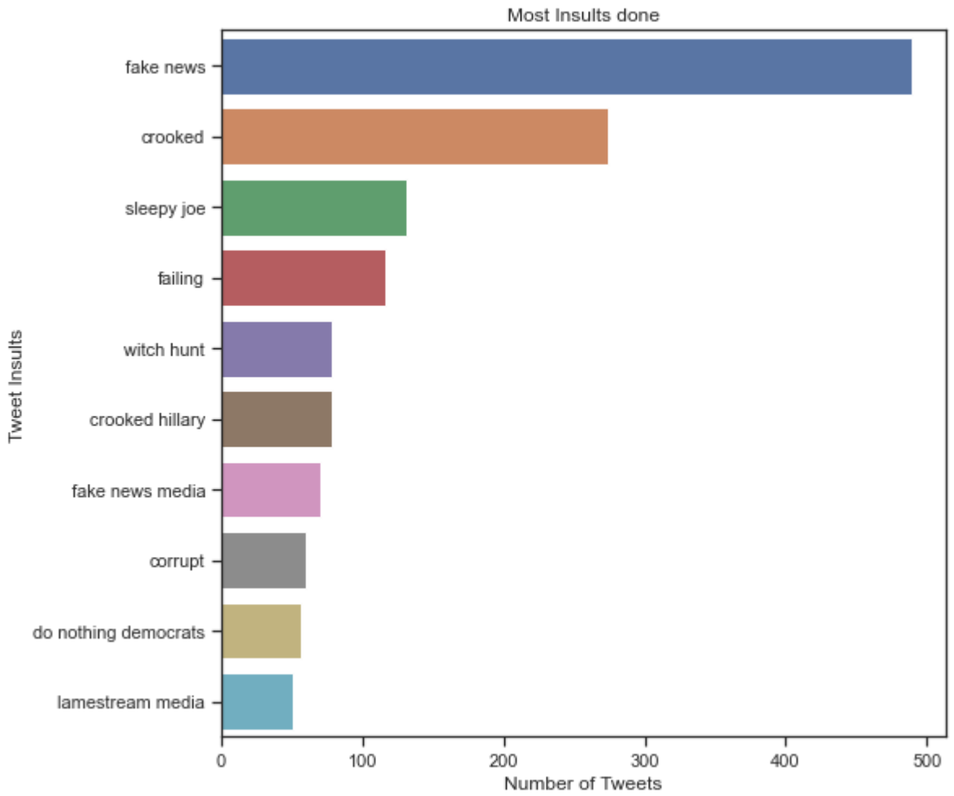


Tweet Insults-per-target

Trump wasn't a hoot-and-scoot type tweeter, he was persistent for some targets, and mostly he was deliberate...

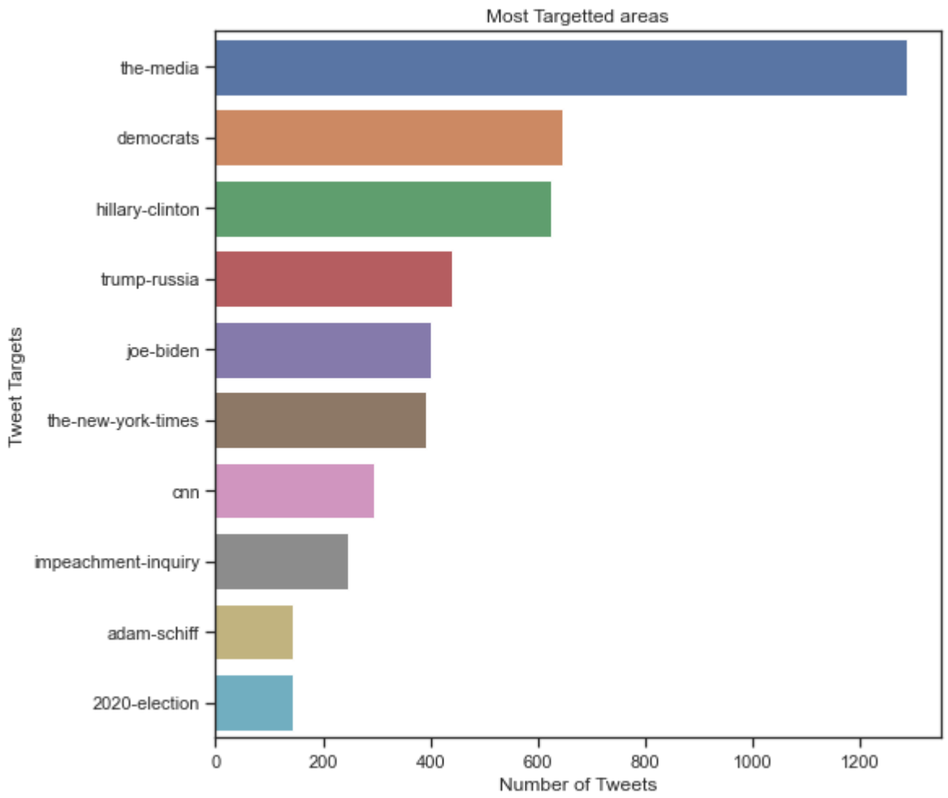
In [23]:
```
# Number of tweets for insults

fig,ax1=plt.subplots(figsize=(8,8))
df['insult']=df['insult'].str.lower()
g=sns.barplot(x=df['insult'].value_counts()[:10],y=df['insult'].value_counts()[:10].index)
g.set_ylabel('Tweet Insults')
g.set_xlabel('Number of Tweets')
g.set_title('Most Insults done')
```

Out[23]:
```
Text(0.5, 1.0, 'Most Insults done')
```



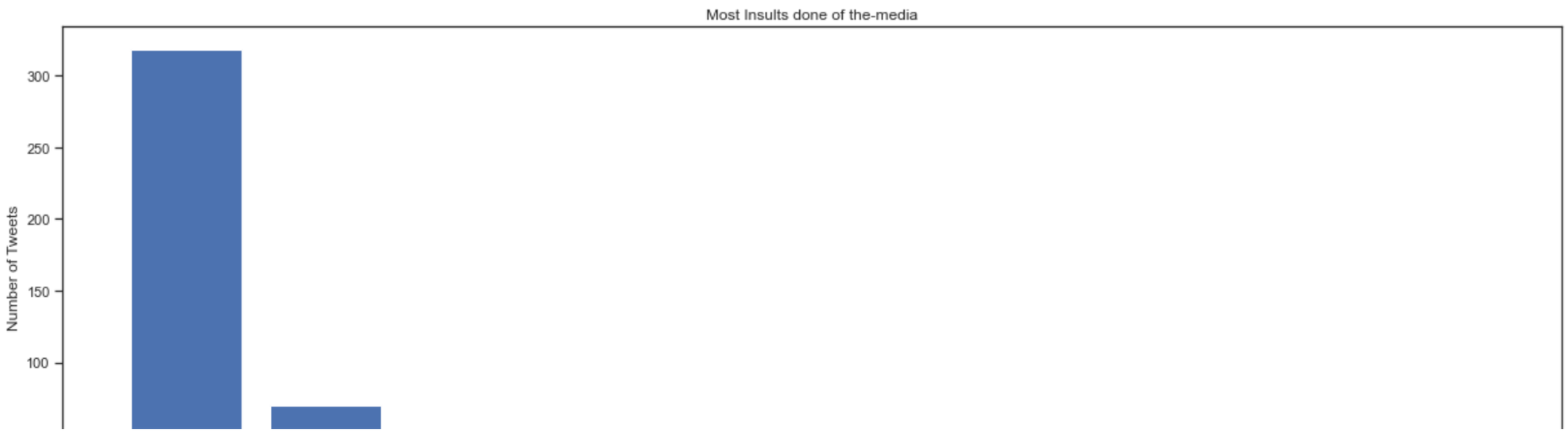
In [24]:
```
# Identify the targets across trump tenure

fig,ax1=plt.subplots(figsize=(8,8))
g=sns.barplot(x=df['target'].value_counts()[:10],y=df['target'].value_counts()[:10].index)
g.set_ylabel('Tweet Targets')
g.set_xlabel('Number of Tweets')
g.set_title('Most Targetted areas')
```

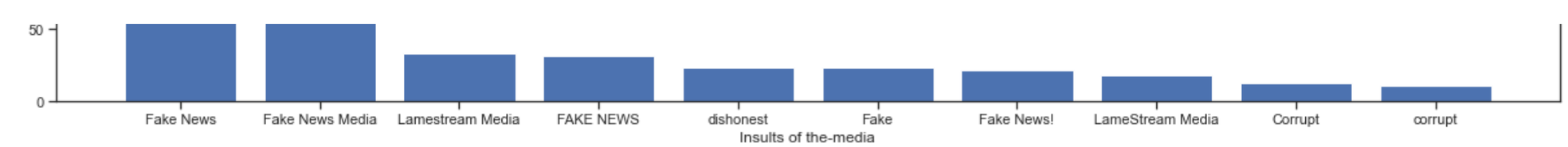
Out[24]:
```
Text(0.5, 1.0, 'Most Targetted areas')
```



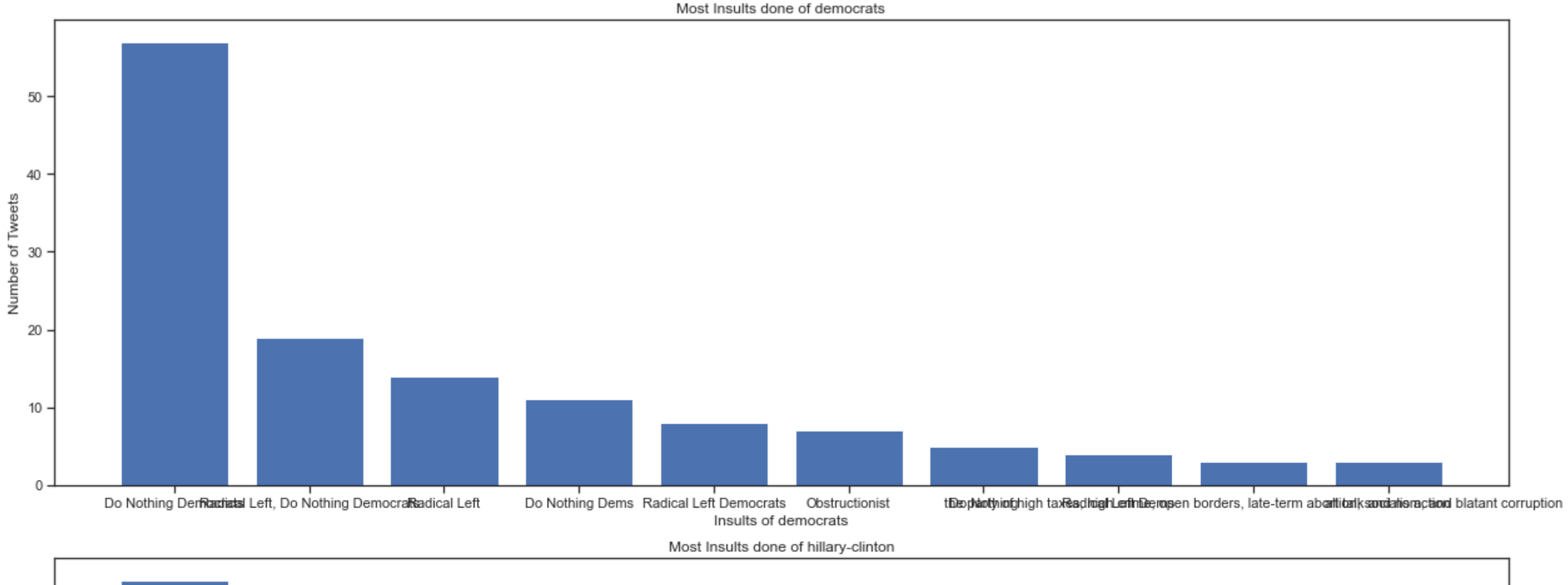
In [19]:

```
# most insults done of the targets
most_targetted=df['target'].value_counts()[:5].index
fig, axs = plt.subplots(5,1, figsize=(18, 30), facecolor='w', edgecolor='k')
fig.subplots_adjust(hspace = .5, wspace=.001)
axs = axs.ravel()
for i in range(0,5):
    x=df[df['target']==most_targetted[i]]['insult'].value_counts()[:10]
    y=df[df['target']==most_targetted[i]]['insult'].value_counts()[:10].index
    axs[i].bar(y,x)
    axs[i].set_xlabel(f'Insults of {most_targetted[i]}')
    axs[i].set_ylabel('Number of Tweets')
    axs[i].set_title(f'Most Insults done of {most_targetted[i]}')
plt.tight_layout()
```

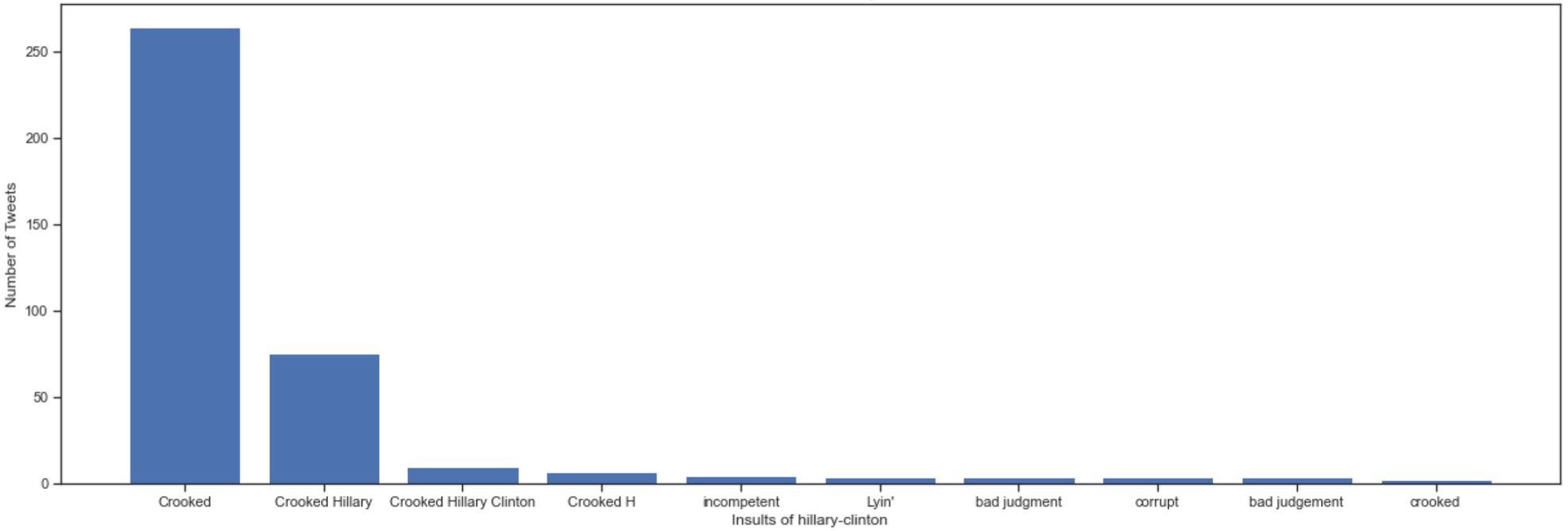




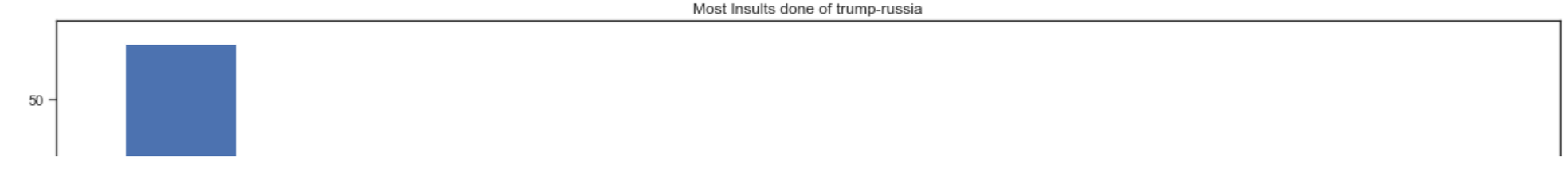
Most Insults done of democrats

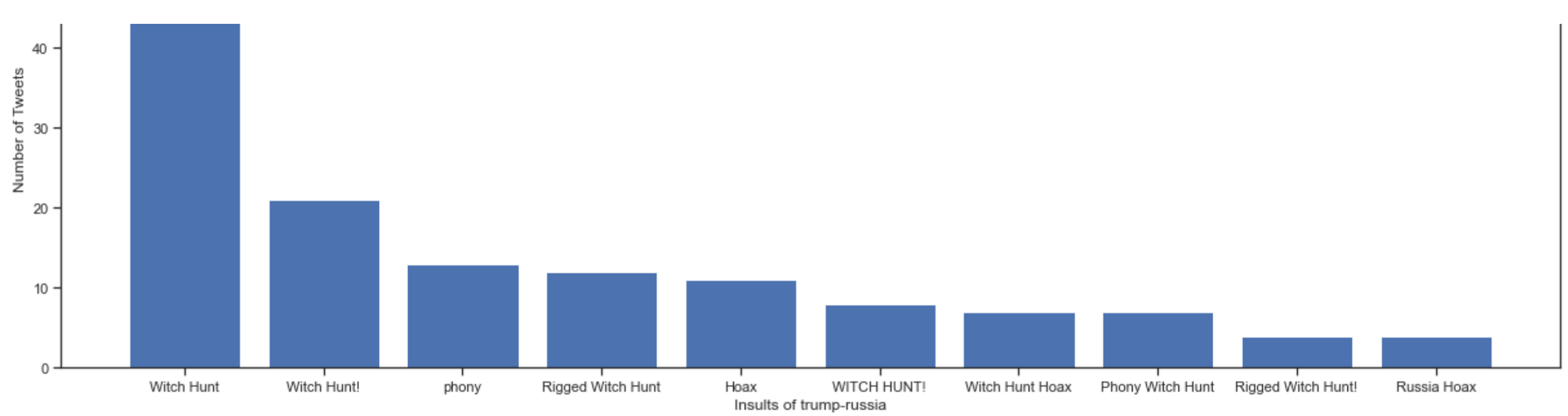


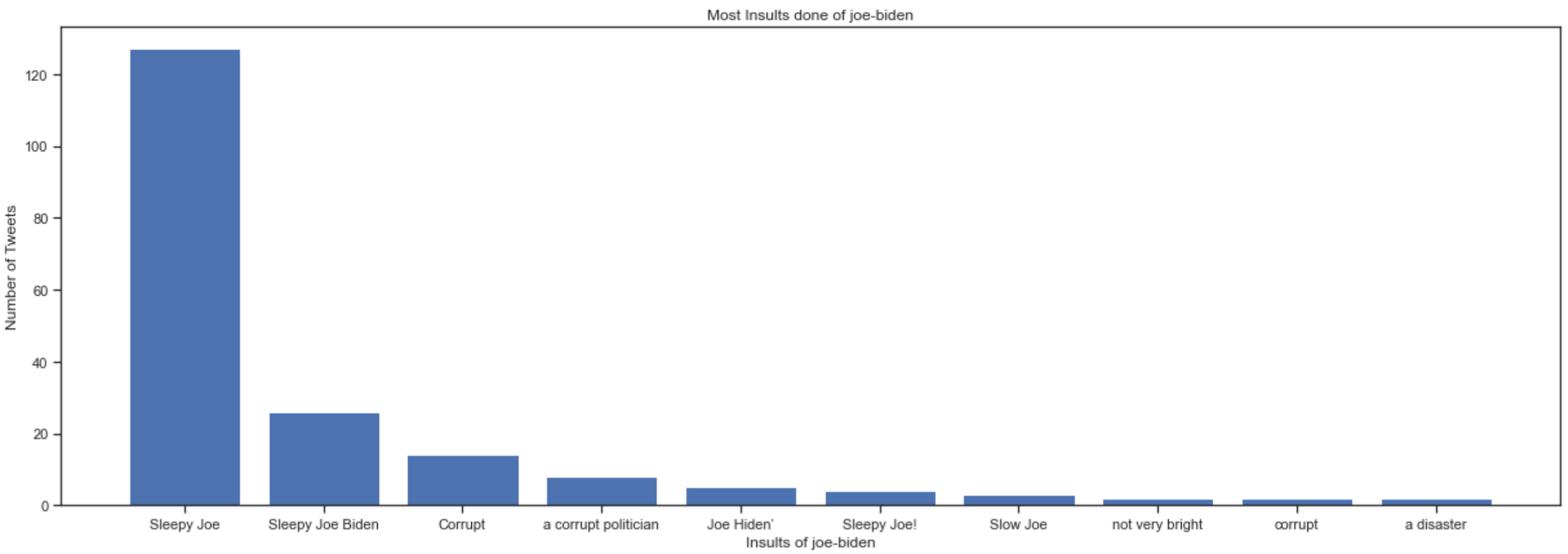
Most Insults done of hillary-clinton



Most Insults done of trump-russia

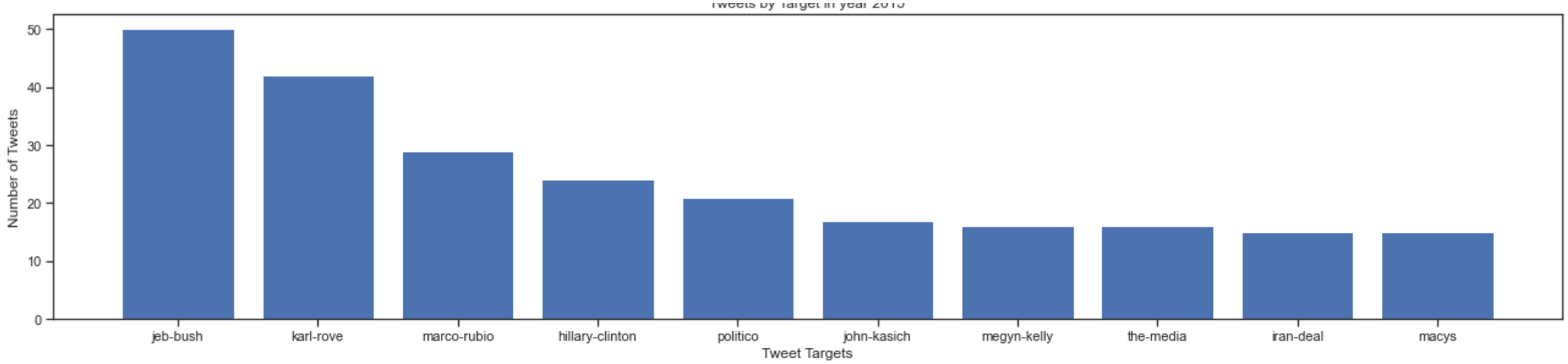


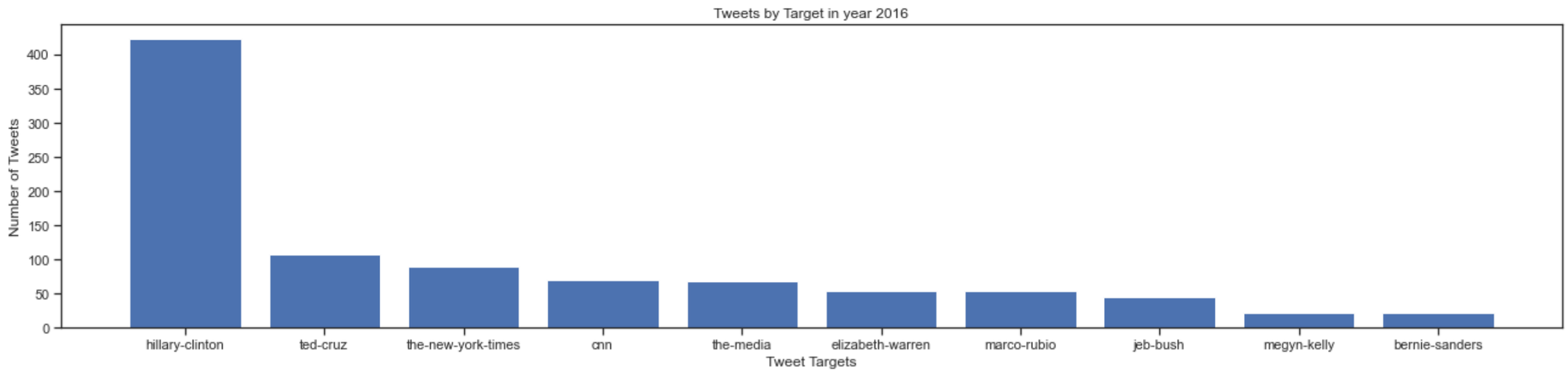


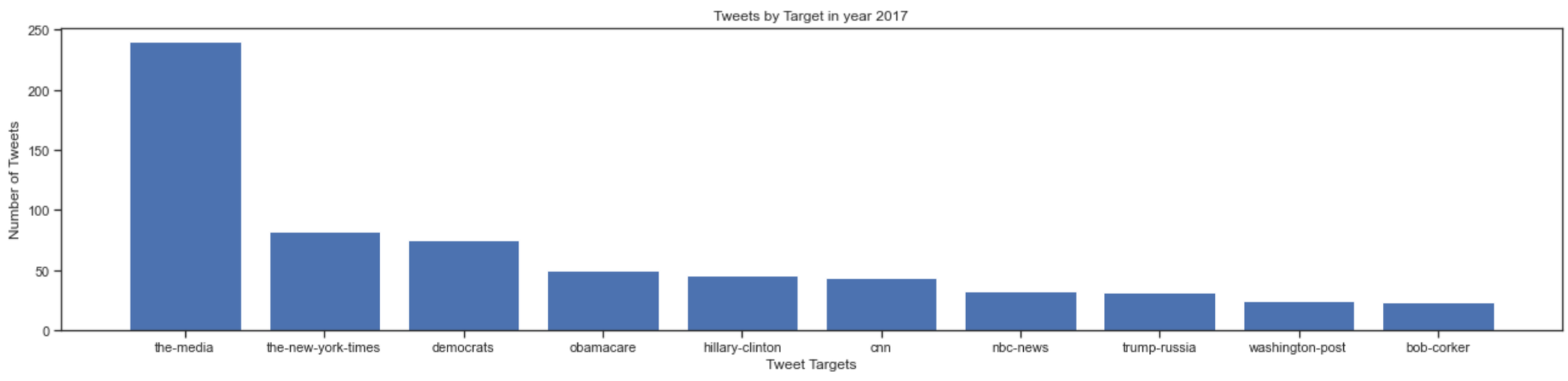


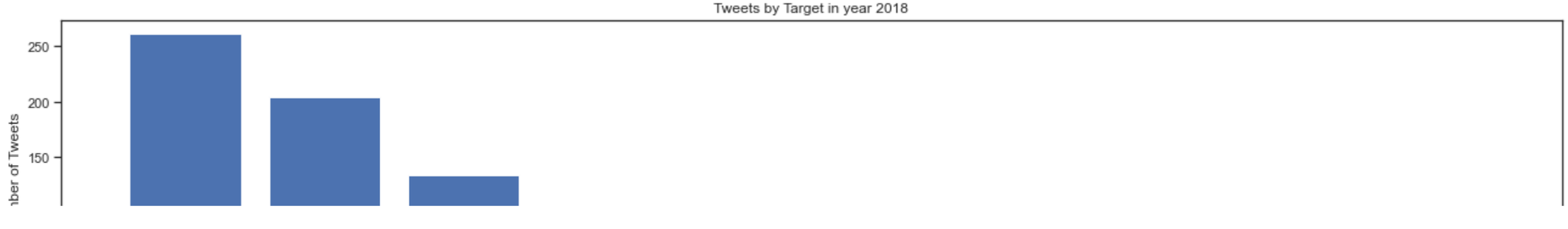
In [33]:

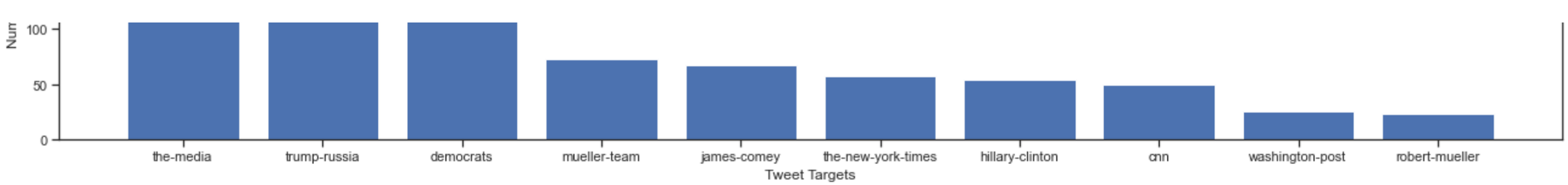
```
# let's see insults per year by targets
year=[]
for x in df['date']:
    x=str(x)
    a=x.split('-')
    year.append(a[0])
df['year']=year
fig, axs = plt.subplots(7,1, figsize=(18, 30), facecolor='w', edgecolor='k')
fig.subplots_adjust(hspace = .5, wspace=.001)
axs = axs.ravel()
for i in range(0,7):
    x=df[df['year']==str(2015+i)]['target'].value_counts()[:10]
    y=df[df['year']==str(2015+i)]['target'].value_counts()[:10].index
    axs[i].bar(y,x)
    axs[i].set_xlabel('Tweet Targets')
    axs[i].set_ylabel('Number of Tweets')
    axs[i].set_title(f'Tweets by Target in year {2015+i}')
plt.tight_layout()
```

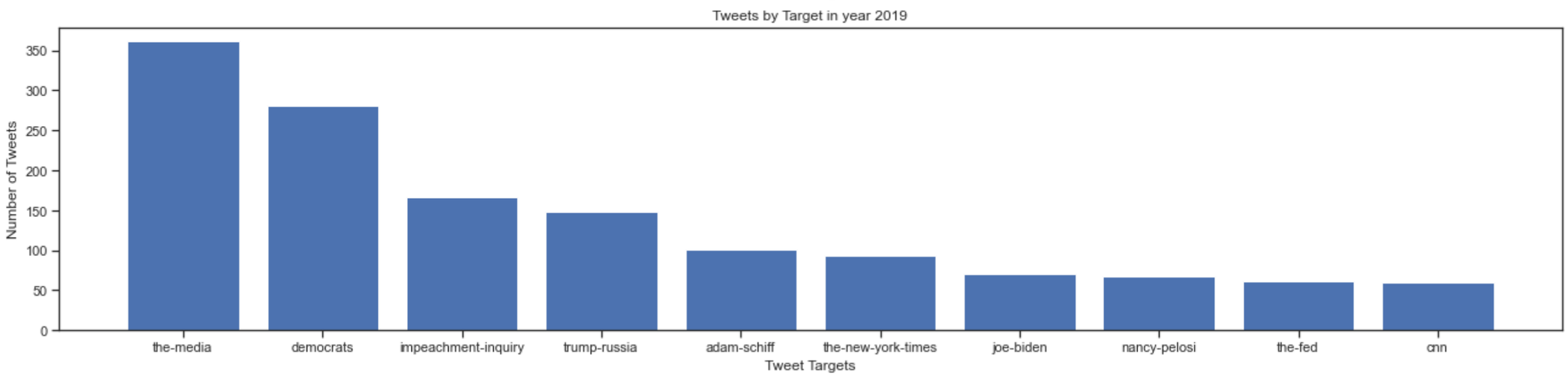

Number of Tweets
0
10
20
30
40
50
jeb-bush
karl-rove
marco-rubio
hillary-clinton
politico
john-kasich
megyn-kelly
the-media
iran-deal
macys
Tweet Targets

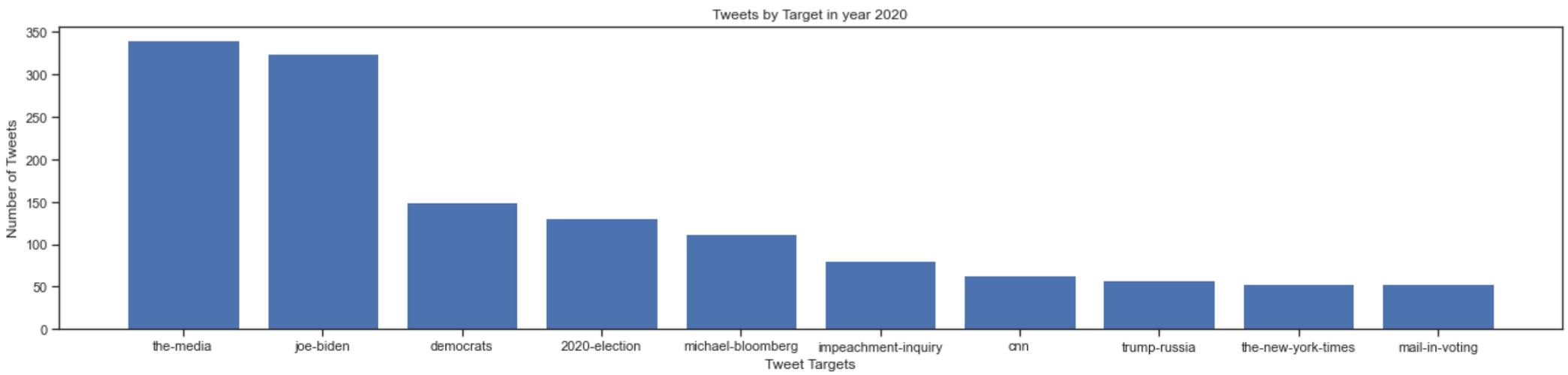

Tweets by Target in year 2016
Number of Tweets
0
50
100
150
200
250
300
350
400
hillary-clinton
ted-cruz
the-new-york-times
cnn
the-media
elizabeth-warren
marco-rubio
jeb-bush
megyn-kelly
bernie-sanders
Tweet Targets

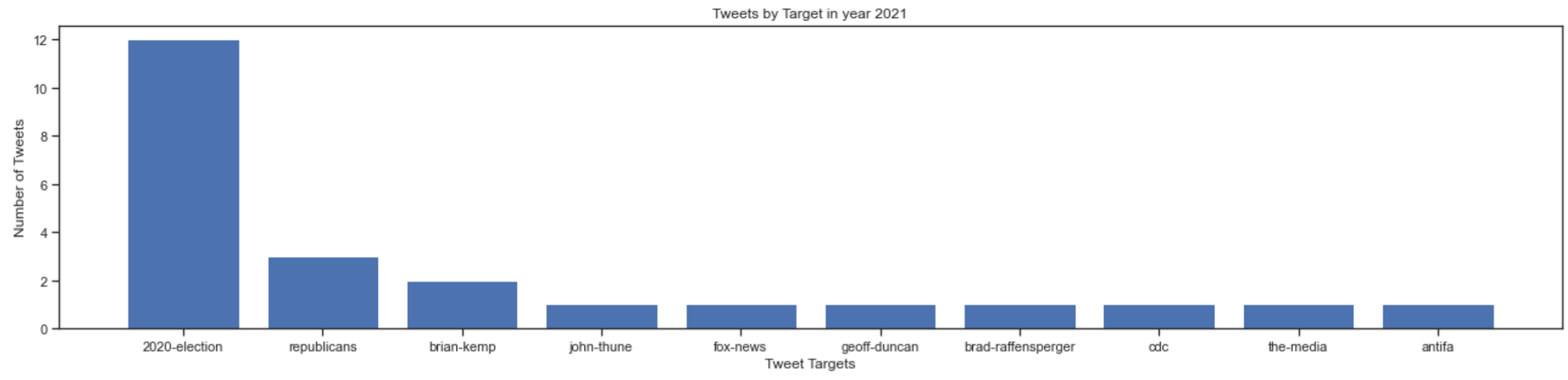

Tweets by Target in year 2017
Number of Tweets
0
50
100
150
200
250
the-media
the-new-york-times
democrats
obamacare
hillary-clinton
cnn
nbc-news
trump-russia
washington-post
bob-corker
Tweet Targets


Tweets by Target in year 2018
150
200
250

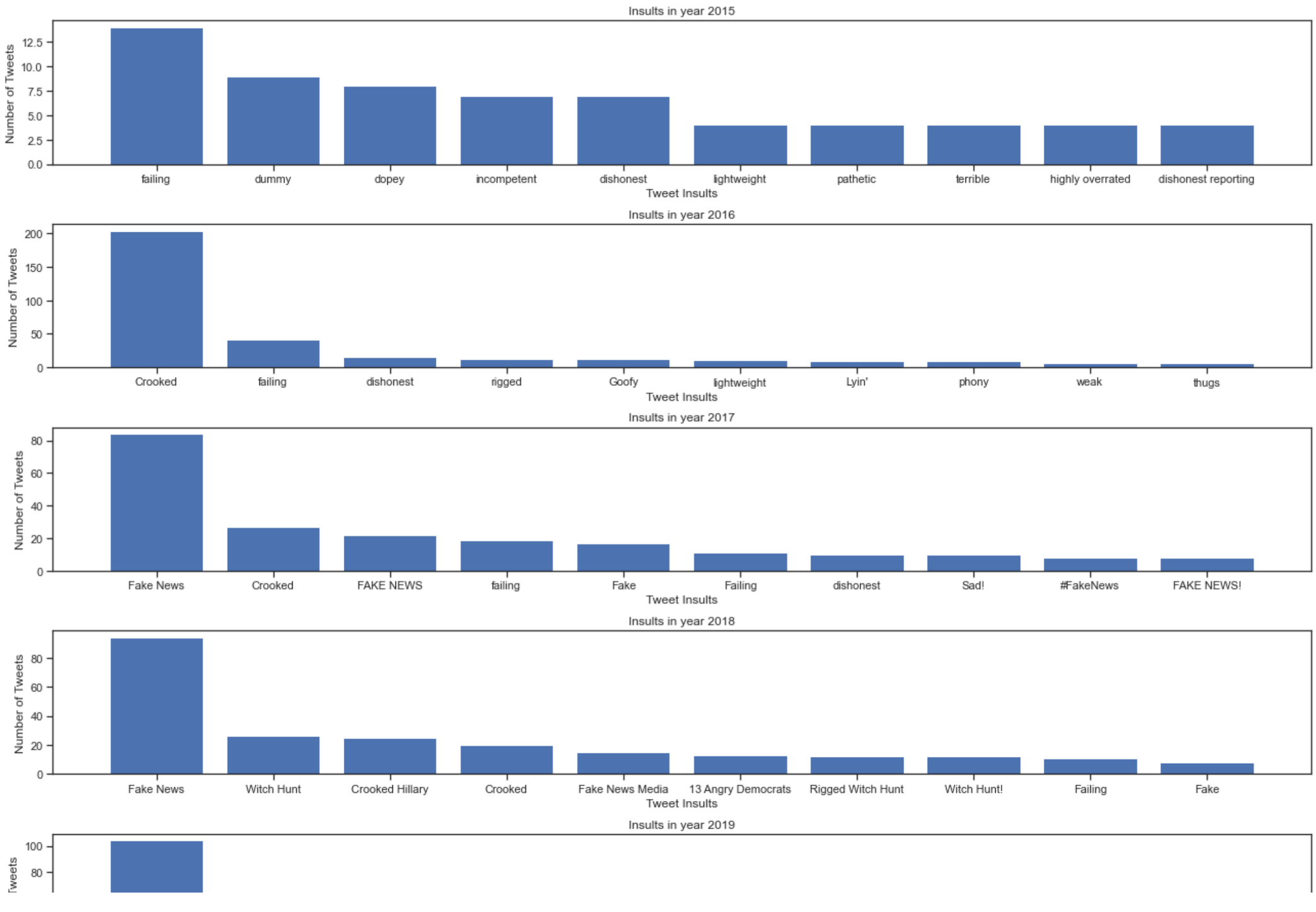

Num
100
50
0
the-media
trump-russia
democrats
mueller-team
james-comey
the-new-york-times
hillary-clinton
cnn
washington-post
robert-mueller
Tweet Targets

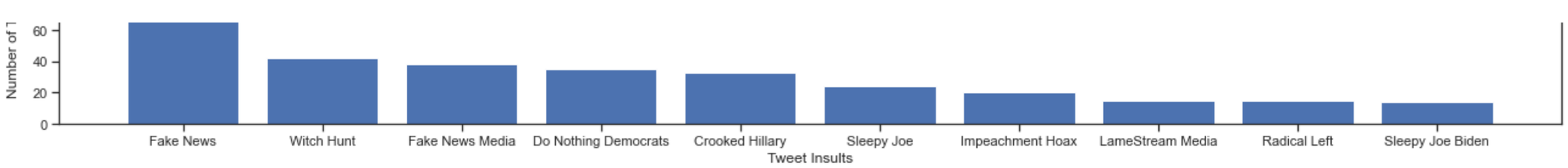

Tweets by Target in year 2019
Number of Tweets
350
300
250
200
150
100
50
0
the-media
democrats
impeachment-inquiry
trump-russia
adam-schiff
the-new-york-times
joe-biden
nancy-pelosi
the-fed
cnn
Tweet Targets

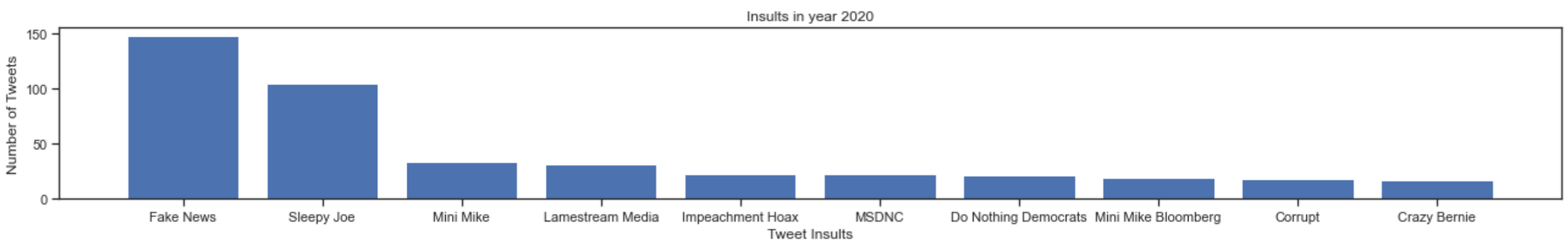

Tweets by Target in year 2020
Number of Tweets
350
300
250
200
150
100
50
0
the-media
joe-biden
democrats
2020-election
michael-bloomberg
impeachment-inquiry
cnn
trump-russia
the-new-york-times
mail-in-voting
Tweet Targets

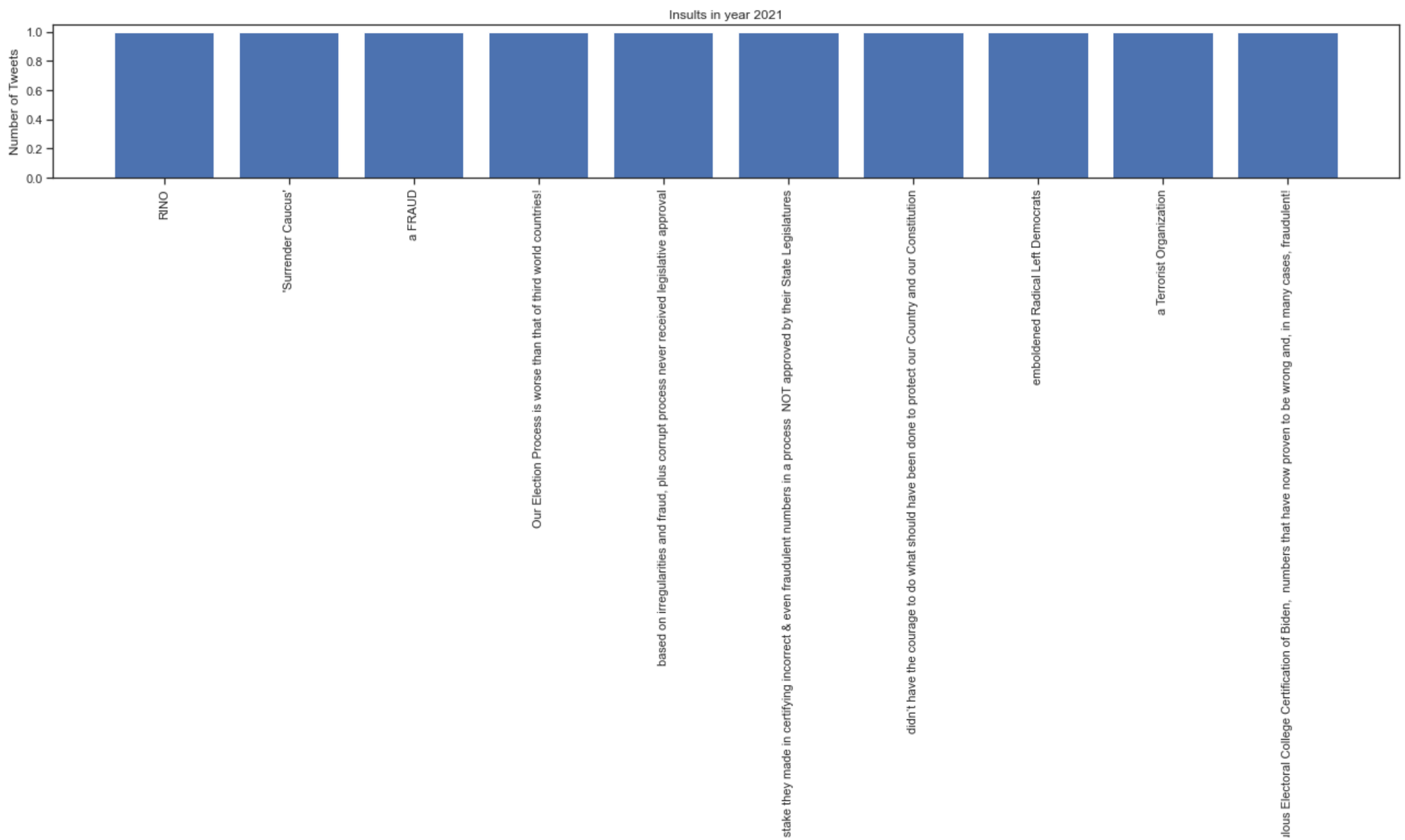

Tweets by Target in year 2021
Number of Tweets
12
10
8
6
4
2
0
2020-election
republicans
brian-kemp
john-thune
fox-news
geoff-duncan
brad-raffensperger
cdc
the-media
antifa
Tweet Targets

In [23]:
```
fig, axs = plt.subplots(7,1, figsize=(18, 30), facecolor='w', edgecolor='k')
fig.subplots_adjust(hspace = .5, wspace=.001)
axs = axs.ravel()
for i in range(0,7):
    x=df[df['year']==str(2015+i)]['insult'].value_counts()[:10]
    y=df[df['year']==str(2015+i)]['insult'].value_counts()[:10].index
    axs[i].bar(y,x)
    axs[i].set_xlabel('Tweet Insults')
    axs[i].set_ylabel('Number of Tweets')
    plt.xticks(rotation=90)
    axs[i].set_title(f'Insults in year {2015+i}')
plt.tight_layout()
```




Number of T
60
40
20
0
Fake News
Witch Hunt
Fake News Media
Do Nothing Democrats
Crooked Hillary
Sleepy Joe
Impeachment Hoax
LameStream Media
Radical Left
Sleepy Joe Biden
Tweet Insults


Insults in year 2020
Number of Tweets
150
100
50
0
Fake News
Sleepy Joe
Mini Mike
Lamestream Media
Impeachment Hoax
MSDNC
Do Nothing Democrats
Mini Mike Bloomberg
Corrupt
Crazy Bernie
Tweet Insults


Insults in year 2021
Number of Tweets
1.0
0.8
0.6
0.4
0.2
0.0
RINO
'Surrender Caucus'
a FRAUD
Our Election Process is worse than that of third world countries!
based on irregularities and fraud, plus corrupt process never received legislative approval
stake they made in certifying incorrect & even fraudulent numbers in a process NOT approved by their State Legislatures
didn't have the courage to do what should have been done to protect our Country and our Constitution
emboldened Radical Left Democrats
a Terrorist Organization
lous Electoral College Certification of Biden, numbers that have now proven to be wrong and, in many cases, fraudulent!

In [21]:

```
df_target = pd.DataFrame(df.groupby('target')['insult'].count().reset_index())
df_target.columns=['target','count']
df_target = df_target.sort_values(['count'],ascending=False)
df_target
```

Out[21]:

	target	count
770	the-media	1287
223	democrats	647
331	hillary-clinton	625
802	trump-russia	441
394	joe-biden	402
...	...	...
641	puerto-rico-aid-relief	1
288	frank-bruni	1
290	frank-vandersloot	1
293	french-wine	1
865	yoel-roth	1

866 rows × 2 columns

==========

#3a. -- Workings Detail... Ref Section Heading: [C.1.1] Exploratory Analysis

==========

Exploratory Data Analysis... iteration

In [47]:

```
df['date'] = pd.to_datetime(df.date, errors = 'ignore')
df.head()
```

Out[47]:

	unnamed: 0	date	target	insult	tweet	year	month	week
0	1	2014-10-09	thomas-frieden	fool	Can you believe this fool, Dr. Thomas Frieden ...	2014	10	41
1	2	2014-10-09	thomas-frieden	dope	Can you believe this fool, Dr. Thomas Frieden ...	2014	10	41
2	3	2015-06-16	politicians	all talk and no action	Big time in U.S. today - MAKE AMERICA GREAT AG...	2015	6	25
3	4	2015-06-24	ben-cardin	it's politicians like cardin that have destroy...	Politician @SenatorCardin didn't like that I s...	2015	6	26
4	5	2015-06-24	neil-young	total hypocrite	For the nonbeliever, here is a photo of @Neily...	2015	6	26

In [48]:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import re

from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator

import nltk
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

stop_words = nltk.corpus.stopwords.words('english')
stop_words.extend(['co', 'wa', 'ha'])
```

In [49]:

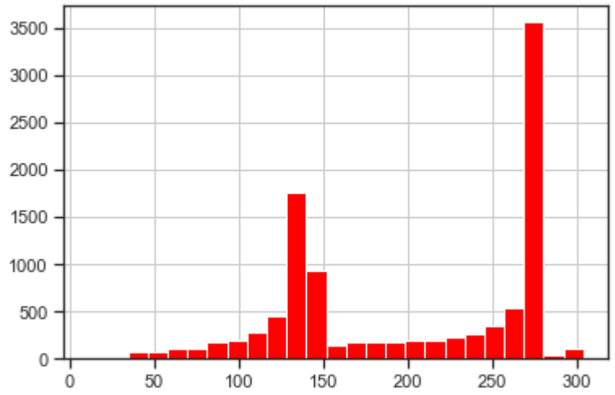
```
# preprocessing , data cleansing
# Initialize the Lemmatizer and Whitespace Tokenizer
w_tokenizer = nltk.tokenize.WhitespaceTokenizer()
lemmatizer = nltk.stem.WordNetLemmatizer()

# lemmatize text column by using a lemmatize function
def lemmatize_text(text):
    return [lemmatizer.lemmatize(w) for w in w_tokenizer.tokenize(text.lower())]

def cleantext(txt):
    # A bit of cleaning
    txt = txt.lower()
    # remove special characters from text column
    txt = re.sub(r'\W', ' ', txt)
    #Remove twitter handlers
    txt = re.sub('@[^\\s]+','', txt)
    #Remove digits
    txt = re.sub(r'\d+',' ', txt)
    # remove urls spaces with single space
    txt = re.sub(r"\"?http\\S+", ' ', txt)
    # remove urls spaces with single space
    txt = re.sub(r"\"?www\\S+", ' ', txt)
    #remove all single characters
    txt = re.sub(r'\\s+[a-zA-Z]\\s+', ' ', txt)
    # remove multiple spaces with single space
    txt = re.sub(r'\\s+', ' ', txt)
    # Lemmatizes
    txt = lemmatize_text(txt)
    tokens = []
    for w in txt:
        if w not in stop_words:
            tokens.append(w)

    return tokens
```

In [50]:

```
#length of the tweets
seq_length = [len(i) for i in df['tweet']]
pd.Series(seq_length).hist(bins=25, color='red')
```

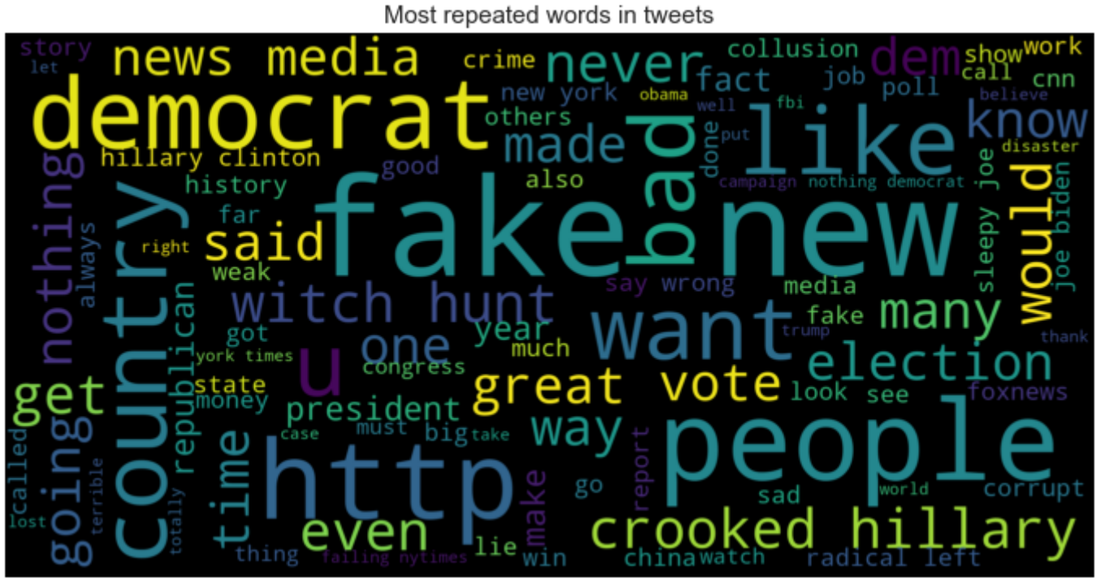
Out[50]:

```
<AxesSubplot:>
```



In [51]:

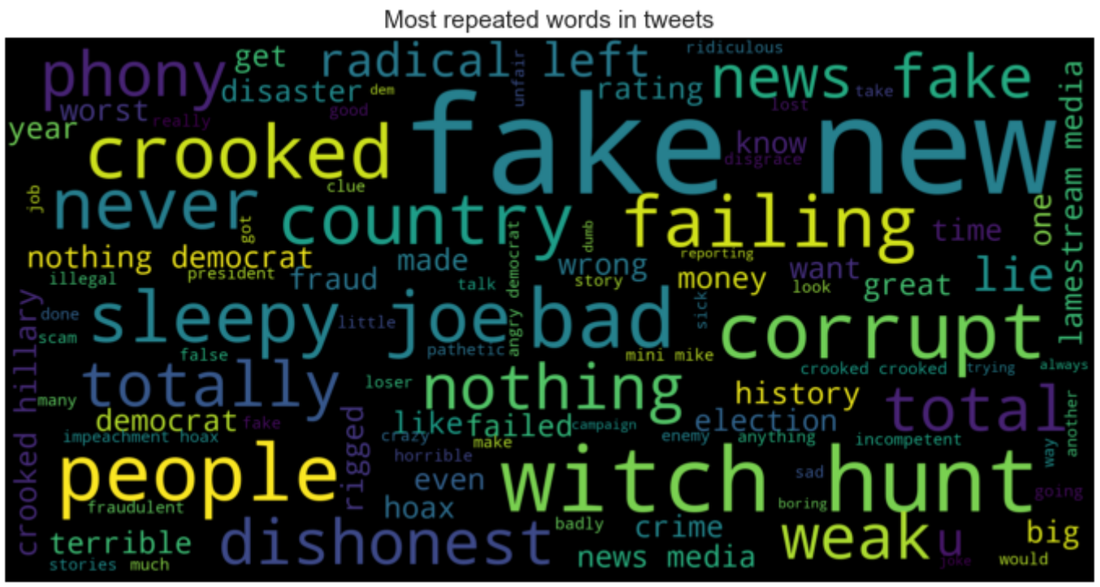
```
# Get a string of tweets
tweet_text = ",".join(tw.lower() for tw in df.tweet)

# Create and generate a word cloud image:
wordcloud = WordCloud(max_font_size=50,
                      max_words=100,
                      stopwords=stop_words,
                      scale=5,
                      background_color="black").generate(tweet_text)

plt.figure(figsize=(12,8))
plt.title('Most repeated words in tweets',fontsize=15)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



In [52]:

```
# Get a string of insults
insult_text = ",".join(tw.lower() for tw in df.insult)

# Create and generate a word cloud image:
wordcloud = WordCloud(max_font_size=50,
                      max_words=100,
                      stopwords=stop_words,
                      scale=5,
                      background_color="black").generate(insult_text)
```

```
plt.figure(figsize=(12,8))
plt.title('Most repeated words in tweets',fontsize=15)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



==========

#3b. -- Workings Detail... Ref Section Heading: [C.1.2] Sentiment Analysis - VADER

==========

In [53]:
```
analyzer = SentimentIntensityAnalyzer()
```

In [54]:
```
df['neg'] = df['tweet'].apply(lambda x:analyzer.polarity_scores(x)['neg'])
df['neu'] = df['tweet'].apply(lambda x:analyzer.polarity_scores(x)['neu'])
df['pos'] = df['tweet'].apply(lambda x:analyzer.polarity_scores(x)['pos'])
df['compound'] = df['tweet'].apply(lambda x:analyzer.polarity_scores(x)['compound'])
df
```

Out[54]:

	unnamed: 0	date	target	insult	tweet	year	month	week	neg	neu	pos	compound
0	1	2014-10-09	thomas-frieden	fool	Can you believe this fool, Dr. Thomas Frieden ...	2014	10	41	0.119	0.881	0.000	-0.5228
1	2	2014-10-09	thomas-frieden	dope	Can you believe this fool, Dr. Thomas Frieden ...	2014	10	41	0.119	0.881	0.000	-0.5228
2	3	2015-06-16	politicians	all talk and no action	Big time in U.S. today - MAKE AMERICA GREAT AG...	2015	6	25	0.075	0.750	0.175	0.6027
3	4	2015-06-24	ben-cardin	it's politicians like cardin that have destroy...	Politician @SenatorCardin didn't like that I s...	2015	6	26	0.208	0.627	0.165	-0.2755
4	5	2015-06-24	neil-young	total hypocrite	For the nonbeliever, here is a photo of @Neily...	2015	6	26	0.000	1.000	0.000	0.0000
...	...	...	...	...	...	...	...	...	...	...	...	...
10355	10356	2021-01-06	2020-election	many states want to decertify the mistake they...	If Vice President @Mike_Pence comes through fo...	2021	1	1	0.149	0.759	0.092	-0.4809
10356	10357	2021-01-06	2020-election	based on irregularities and fraud, plus corrup...	States want to correct their votes, which they...	2021	1	1	0.106	0.734	0.160	0.5047
10357	10358	2021-01-06	2020-election	our election process is worse than that of thi...	They just happened to find 50,000 ballots late...	2021	1	1	0.275	0.725	0.000	-0.8439
10358	10359	2021-01-06	2020-election	a fraud	The States want to redo their votes. They foun...	2021	1	1	0.220	0.616	0.164	-0.4261

	unnamed: 0	date	target	insult	tweet	year	month	week	neg	neu	pos	compound
10359	10360	2021-01-06	chuck-todd	sleepy eyes, sad to watch!	Sleepy Eyes Chuck Todd is so happy with the fa...	2021	1	1	0.182	0.615	0.203	0.1860

10358 rows × 12 columns

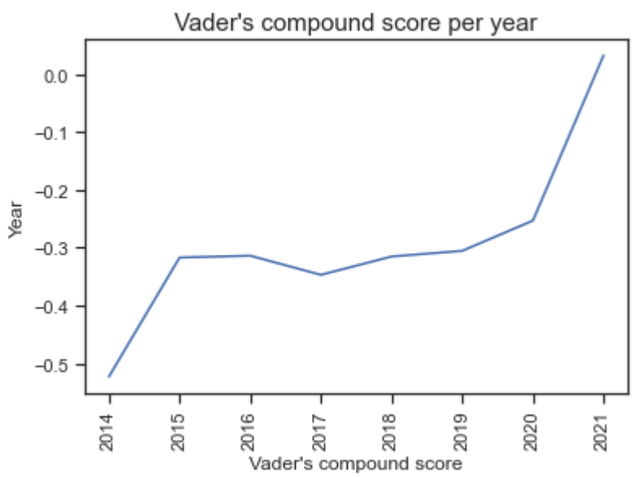
In [55]:
```
yearperiod = df.date.dt.to_period("Y")
ygroup = df.groupby(yearperiod)
ygroup.mean()
```

Out[55]:

date	unnamed: 0	month	week	neg	neu	pos	compound
2014	1.500000	10.000000	41.0	0.119000	0.881000	0.000000	-0.522800
2015	381.000000	9.734478	41.120211	0.195433	0.717597	0.086946	-0.316869
2016	1529.000000	5.814815	23.545809	0.188346	0.731827	0.079840	-0.313764
2017	2858.789991	7.127793	29.208222	0.195811	0.713366	0.090835	-0.346739
2018	4307.000000	6.867192	27.960608	0.174020	0.729274	0.096737	-0.315157
2019	6408.074639	6.923711	28.309691	0.166767	0.739916	0.093307	-0.305331
2020	8977.500000	6.818584	28.253687	0.167607	0.736382	0.096012	-0.253100
2021	10347.000000	1.000000	27.962963	0.121185	0.759407	0.119333	0.032367

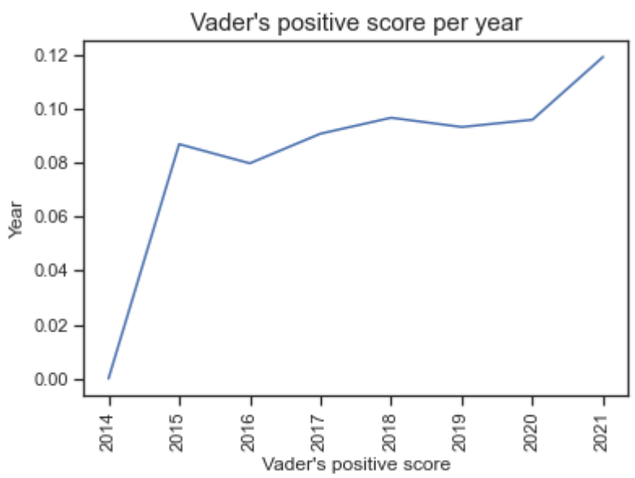
In [56]:
```
#df = df.sort_values('date', ascending=True)
plt.plot(np.unique(yearperiod.values.astype(str)), ygroup.mean()['compound'])
plt.title('Vader\'s compound score per year',fontsize=15)
plt.xlabel('Vader\'s compound score')
plt.ylabel('Year')
plt.xticks(rotation='vertical')
```

Out[56]:
```
([0, 1, 2, 3, 4, 5, 6, 7],
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, '')])
```



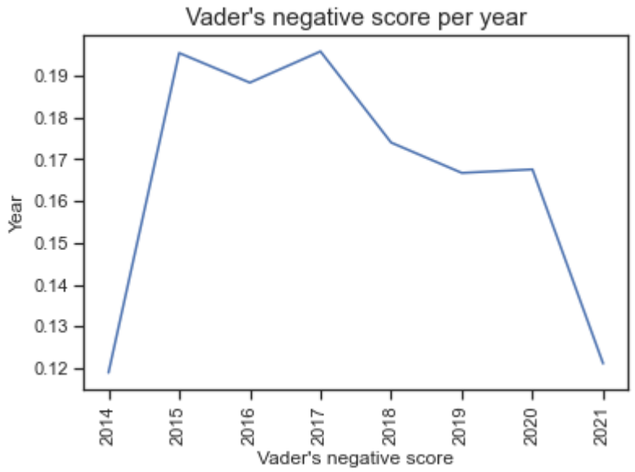
In [57]:
```
#df = df.sort_values('date', ascending=True)
plt.plot(np.unique(yearperiod.values.astype(str)), ygroup.mean()['pos'])
plt.title('Vader\'s positive score per year',fontsize=15)
plt.xlabel('Vader\'s positive score')
plt.ylabel('Year')
plt.xticks(rotation='vertical')
```

Out[57]:
```
([0, 1, 2, 3, 4, 5, 6, 7],
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, '')])
```



In [58]:
```
#df = df.sort_values('date', ascending=True)
plt.plot(np.unique(yearperiod.values.astype(str)), ygroup.mean()['neg'])
plt.title('Vader\'s negative score per year',fontsize=15)
plt.xlabel('Vader\'s negative score')
plt.ylabel('Year')
plt.xticks(rotation='vertical')
```

Out[58]:
```
([0, 1, 2, 3, 4, 5, 6, 7],
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, '')])
```



NLP Sentiment Analysis *(using VADER)... an iteration*

In [105…

```
import vaderSentiment
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
readSentiment = SentimentIntensityAnalyzer()
def getSentiment(phrase):
  s = readSentiment.polarity_scores(phrase)
  if s['compound'] <= -0.05:
    sentiment = 0
  elif s['compound'] >= 0.05:
    sentiment = 1
  else:
    sentiment = 2
  return sentiment, s
```

In [106…

```
phrases = ["I love the team and how they played last night 💘" ,
"What a fine day I am having today :-) :-)",
"I am laughing like crazy lol",
"He was not very good at the play",
"To be or not to be",
"He is kinda bored",
]
sentiments = ['Negative', 'Positive', 'Neutral']
for txt in phrases:
  print(sentiments[getSentiment(txt)[0]], ' - ', txt)
```

```
Positive  -  I love the team and how they played last night 💘
Positive  -  What a fine day I am having today :-) :-)
Positive  -  I am laughing like crazy lol
Negative  -  He was not very good at the play
Neutral  -  To be or not to be
Negative  -  He is kinda bored
```

In [107…

```
f'Qty rows: {df.shape[0]}'
```

Out[107…

```
'Qty rows: 10358'
```

In [108…

```
df['sentimentVader'] = df.tweet.apply(lambda x: sentiments[getSentiment(x)[0]])
df['sentimentVader'].sample(10)
```

Out[108…

```
1580    Negative
1157    Negative
7979    Negative
```

```
3793    Negative
6182    Positive
3405    Positive
9187    Positive
1734    Negative
8362    Negative
1914    Negative
Name: sentimentVader, dtype: object
```

In [109…

```
sentiment = df.groupby('sentimentVader')['sentimentVader'].count()
sentiment
```

Out[109…

```
sentimentVader
Negative    7046
Neutral      381
Positive    2931
Name: sentimentVader, dtype: int64
```

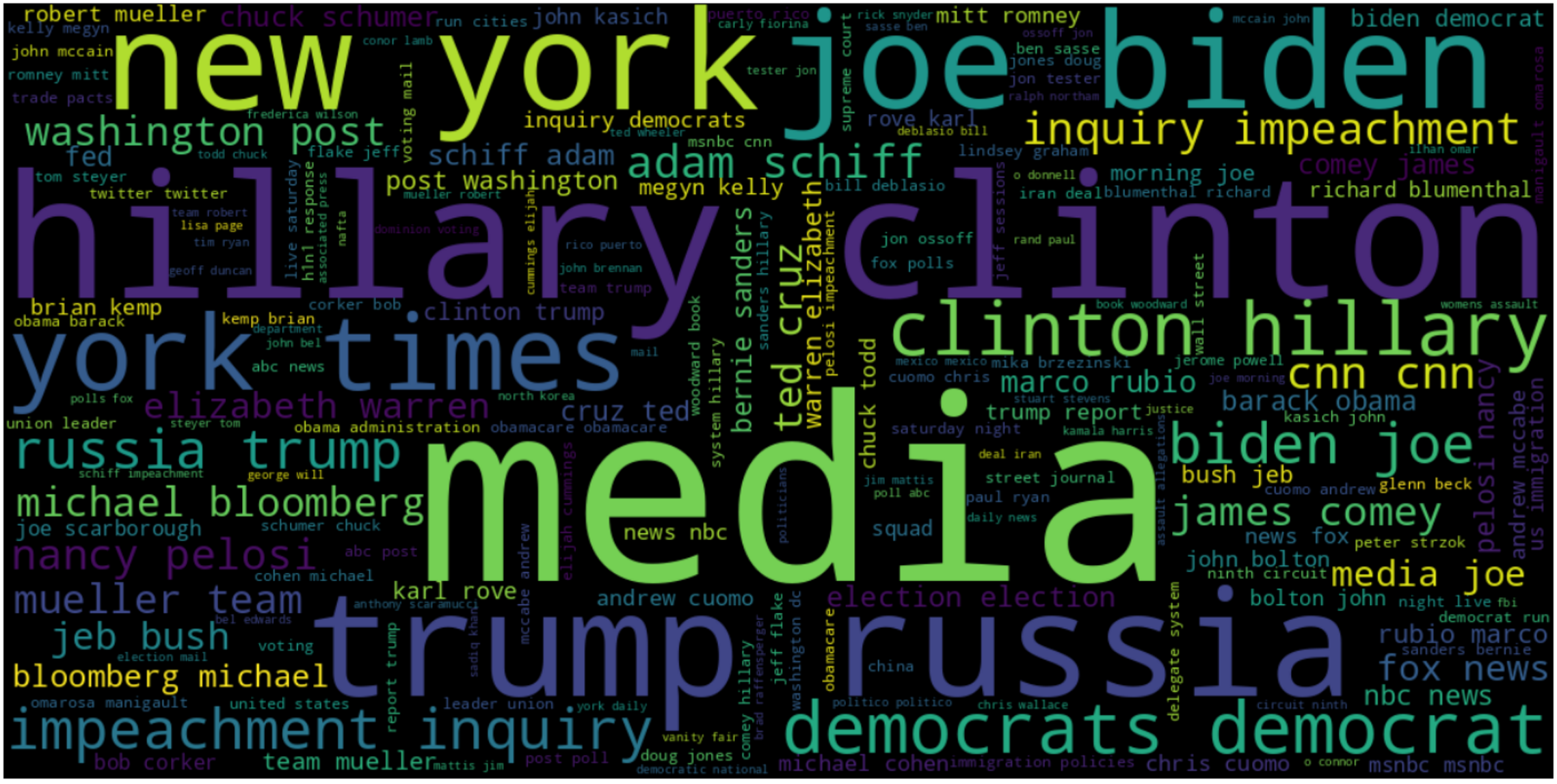
==========

#3c. -- Workings Detail... Ref Section Heading: [C.1.3] Sentiment Analysis - WordCloud

==========

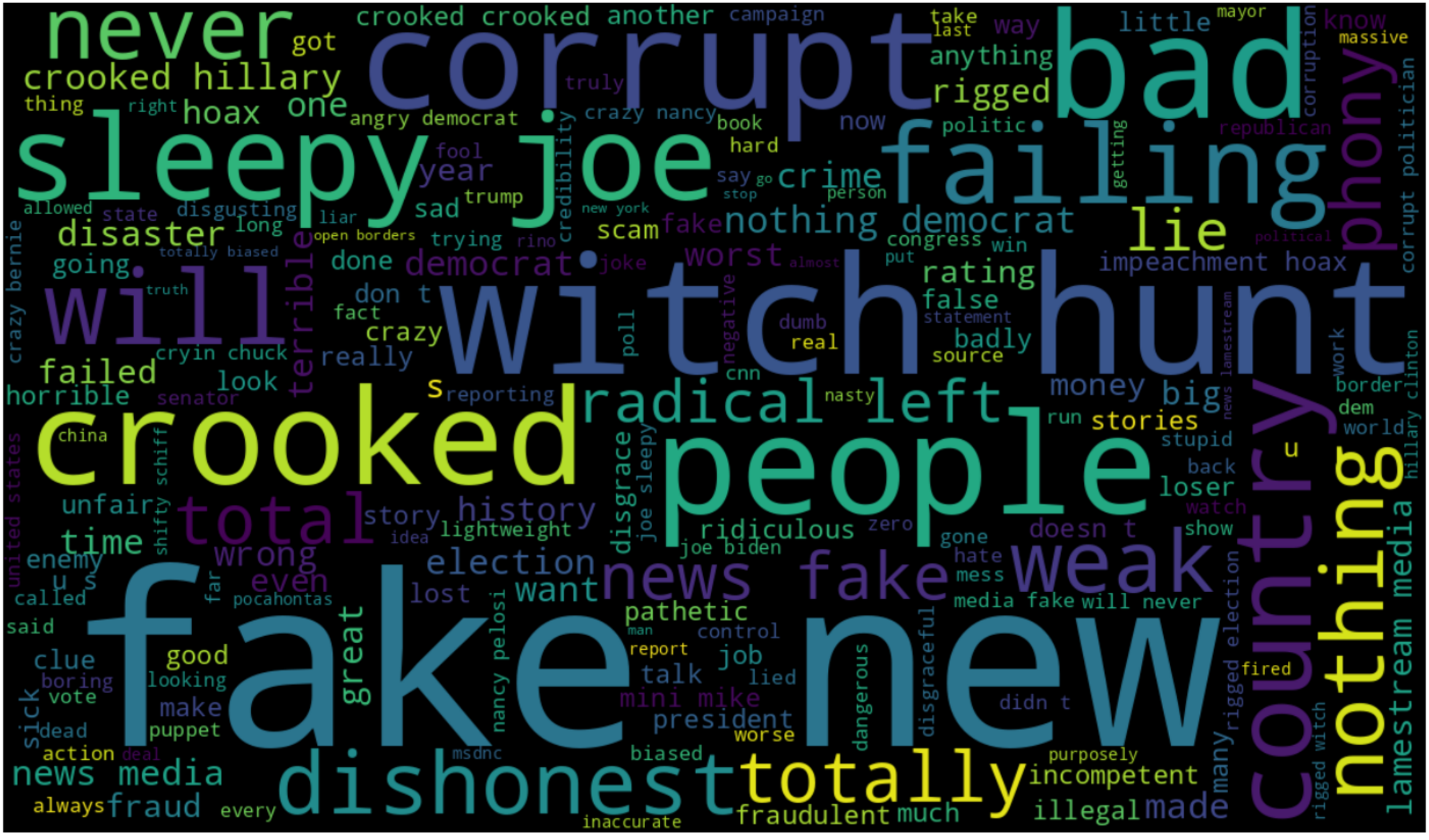
In [25]:

```
#show Wordcloud for the dataset
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator

target_text = " ".join(df.target)
wordcloud = WordCloud(width=1200, height=600).generate(text=target_text)
plt.figure(figsize=(25,18))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```



In [26]:

```
insult_text = " ".join(df.insult)
wordcloud = WordCloud(width=1200, height=700).generate(text=insult_text)
plt.figure(figsize=(25,18))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```



WordCloud implemenation #2

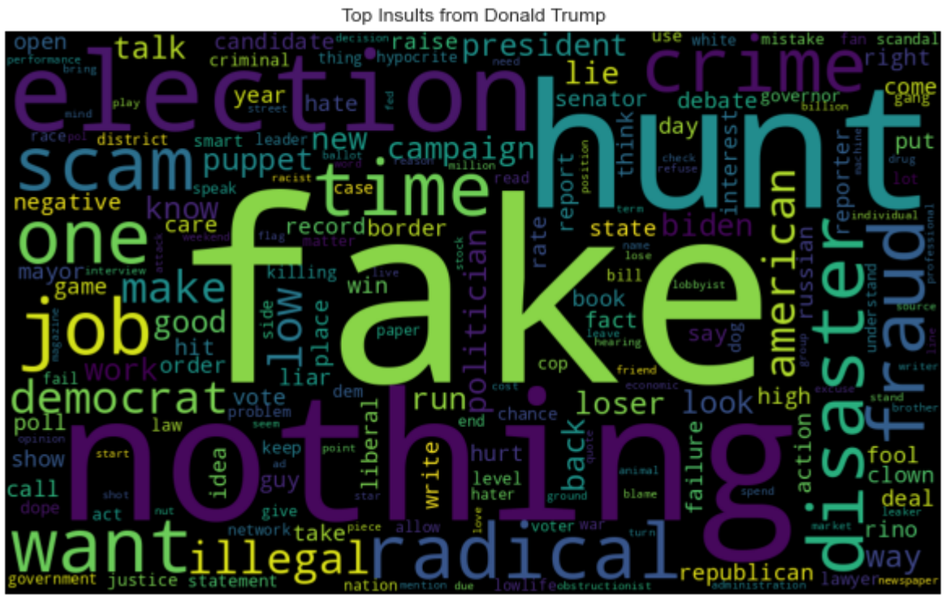
In [100…

```
from sklearn.feature_extraction.text import CountVectorizer
import matplotlib.pyplot as plt
from wordcloud import WordCloud, STOPWORDS
stopwords = set(STOPWORDS)

def myWordCloud(data, title, width = 1000, height = 600,):
  wordcloud = WordCloud(width = width, height = height,
                background_color ='black',
                stopwords = stopwords,
                min_font_size = 10).generate(' '.join(data))

  # plot the WordCloud image
```

```
  plt.figure(figsize = (8, 8), facecolor = None)
  plt.imshow(wordcloud)
  plt.axis("off")
  plt.tight_layout(pad = 0)
  plt.title(title)
  plt.show()
```

In [101…

```
cv = CountVectorizer()
matriz = cv.fit_transform(df.insult)
contaPalavra = pd.DataFrame(cv.get_feature_names(), columns={'Word'})
contaPalavra['Counter'] = matriz.sum(axis=0).tolist()[0]
contaPalavra = contaPalavra.sort_values('Counter', ascending=False).reset_index(drop=True)

myWordCloud(contaPalavra.Word, f'Top Insults from Donald Trump')
```

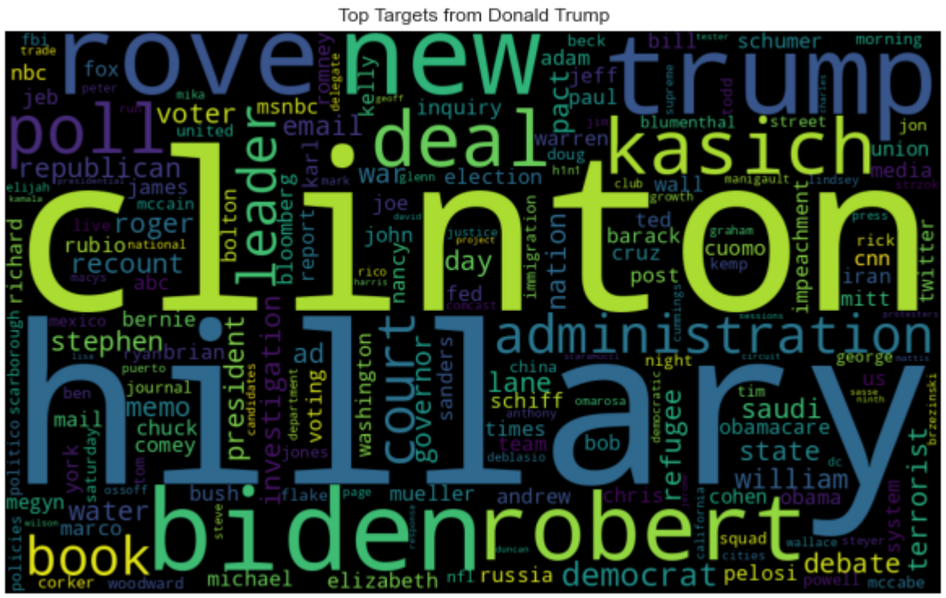


In [102…

```
df.target = df.target.apply(lambda x: x.lower())
targets = df.target.unique().tolist()
df['tokenTarget'] = df.target.apply(lambda x: targets.index(x))
targetsId = df.tokenTarget.unique().tolist()
```

In [104…

```
cv = CountVectorizer()
matriz = cv.fit_transform(df.target)
contaPalavra = pd.DataFrame(cv.get_feature_names(), columns={'Word'})
contaPalavra['Counter'] = matriz.sum(axis=0).tolist()[0]
contaPalavra = contaPalavra.sort_values('Counter', ascending=False).reset_index(drop=True)

myWordCloud(contaPalavra.Word, f'Top Targets from Donald Trump')
```



==========

#3d. -- Workings Detail... Ref Section Heading: [C.2] Latent Dirichlet Allocation (LDA) Modeling

==========

Intention of LDA Modeling is to try answer... What is the core of Trump insults? Can these tweets be programmatically organized into topics? Can patterns be discovered that may be useful for further analysis/modeling?

In [25]:
```
#pip install python-Levenshtein
from gensim.corpora import Dictionary
id2word = Dictionary(tweet_df_token)
texts = tweet_df_token
corpus = [id2word.doc2bow(text) for text in texts]
corpus[:1]
```

Out[25]:
```
[[(0, 1),
  (1, 1),
  (2, 1),
  (3, 1),
  (4, 1),
  (5, 1),
  (6, 1),
  (7, 1),
  (8, 1),
  (9, 1),
  (10, 1),
  (11, 1),
  (12, 1)]]
```

In [26]:
```
from gensim.models import LdaModel
lda_model = LdaModel(corpus=corpus,
                     id2word=id2word,
                     num_topics=20,
                     random_state=100,
                     update_every=1,
                     chunksize=100,
                     passes=10,
                     alpha='auto',
                     per_word_topics=True)
```

In [27]:

```
#LDAvis is a visualization package for LDA Model
#its an interactive visualization
#P.S. --> Consider exploring the Intertopic Distance Map vis-a-vis Top-30 Salient Terms
import pyLDAvis
import pyLDAvis.gensim_models as gensimvis
import matplotlib.pyplot as plt
%matplotlib inline

# Visualize the topics
pyLDAvis.enable_notebook()
vis = gensimvis.prepare(lda_model, corpus, id2word)
vis
```

```
/usr/local/lib/python3.9/site-packages/past/builtins/misc.py:45: DeprecationWarning: the imp module is deprecated in favour of importlib; see the module's documentation for alternative uses
  from imp import reload
/usr/local/lib/python3.9/site-packages/past/builtins/misc.py:45: DeprecationWarning: the imp module is deprecated in favour of importlib; see the module's documentation for alternative uses
  from imp import reload
/usr/local/lib/python3.9/site-packages/past/builtins/misc.py:45: DeprecationWarning: the imp module is deprecated in favour of importlib; see the module's documentation for alternative uses
  from imp import reload
/usr/local/lib/python3.9/site-packages/past/builtins/misc.py:45: DeprecationWarning: the imp module is deprecated in favour of importlib; see the module's documentation for alternative uses
  from imp import reload
/usr/local/lib/python3.9/site-packages/past/builtins/misc.py:45: DeprecationWarning: the imp module is deprecated in favour of importlib; see the module's documentation for alternative uses
  from imp import reload
/usr/local/lib/python3.9/site-packages/past/builtins/misc.py:45: DeprecationWarning: the imp module is deprecated in favour of importlib; see the module's documentation for alternative uses
  from imp import reload
```

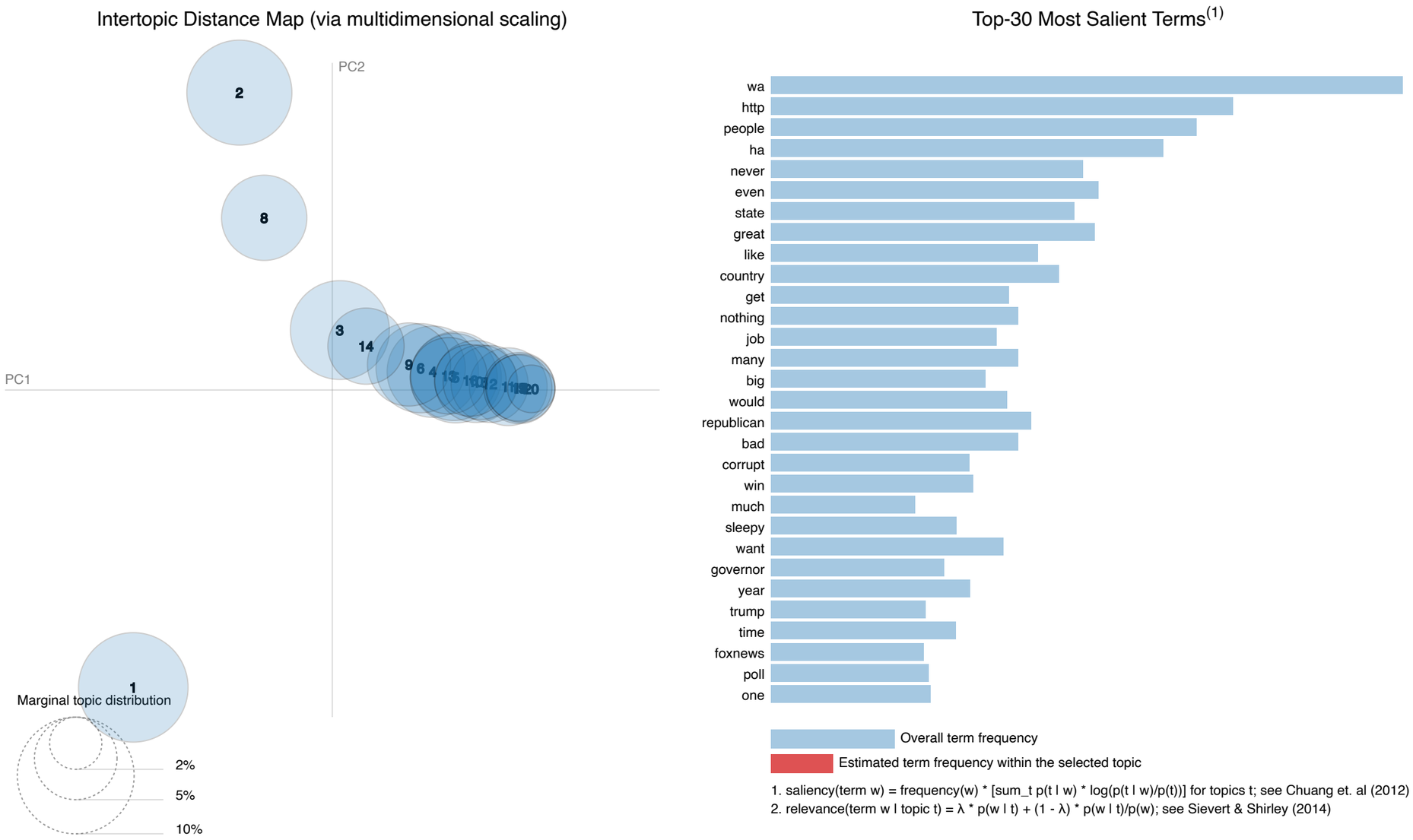
Out[27]:

Selected Topic: 0 [Previous Topic] [Next Topic] [Clear Topic]

Slide to adjust relevance metric:(2)

λ = 1

0.0 0.2 0.4 0.6 0.8 1.0



On the left, the topics are plotted on a 2 dimensional plane representing the distance between each topic. While the right horizontal bar chart represents the words most relevant to each topic. The chart is interactive, allowing you to select specific topics and view the related words for each topic, in hopes of inferring meaning from each topic.

==========

#4a. -- Workings Detail... Ref Section Heading: [D] Model Selection and Evaluation

==========

Random Forest Model

In [60]:
```
Top_targets=most_Insulted.iloc[0:5,0]
Top_5=['the-media', 'democrats','hillary-clinton', 'trump-russia', 'joe-biden']
```

In [61]:
```
df['Top_5'] = df.target.apply(lambda x : 1 if (x  in Top_5 ) else 0)
```

```
y=df['Top_5']
df
```

Out[61]:

	unnamed: 0	date	target	insult	tweet	year	month	week	neg	neu	pos	compound	Top_5
0	1	2014-10-09	thomas-frieden	fool	Can you believe this fool, Dr. Thomas Frieden ...	2014	10	41	0.119	0.881	0.000	-0.5228	0
1	2	2014-10-09	thomas-frieden	dope	Can you believe this fool, Dr. Thomas Frieden ...	2014	10	41	0.119	0.881	0.000	-0.5228	0
2	3	2015-06-16	politicians	all talk and no action	Big time in U.S. today - MAKE AMERICA GREAT AG...	2015	6	25	0.075	0.750	0.175	0.6027	0
3	4	2015-06-24	ben-cardin	it's politicians like cardin that have destroy...	Politician @SenatorCardin didn't like that I s...	2015	6	26	0.208	0.627	0.165	-0.2755	0
4	5	2015-06-24	neil-young	total hypocrite	For the nonbeliever, here is a photo of @Neily...	2015	6	26	0.000	1.000	0.000	0.0000	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
10355	10356	2021-01-06	2020-election	many states want to decertify the mistake they...	If Vice President @Mike_Pence comes through fo...	2021	1	1	0.149	0.759	0.092	-0.4809	0
10356	10357	2021-01-06	2020-election	based on irregularities and fraud, plus corrup...	States want to correct their votes, which they...	2021	1	1	0.106	0.734	0.160	0.5047	0
10357	10358	2021-01-06	2020-election	our election process is worse than that of thi...	They just happened to find 50,000 ballots late...	2021	1	1	0.275	0.725	0.000	-0.8439	0
10358	10359	2021-01-06	2020-election	a fraud	The States want to redo their votes. They foun...	2021	1	1	0.220	0.616	0.164	-0.4261	0
10359	10360	2021-01-06	chuck-todd	sleepy eyes, sad to watch!	Sleepy Eyes Chuck Todd is so happy with the fa...	2021	1	1	0.182	0.615	0.203	0.1860	0

10358 rows × 13 columns

In [62]:
```
#Tokenization using count vectorizer
count_vect = CountVectorizer(ngram_range=(1,1))
token = count_vect.fit_transform(tweet_df_lemma)
token
```

Out[62]:
```
<10358x8977 sparse matrix of type '<class 'numpy.int64'>'
        with 301429 stored elements in Compressed Sparse Row format>
```

In [63]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vect = TfidfVectorizer(ngram_range=(1,1), min_df=0.05, max_df=0.95)
tfidf_vect
```

Out[63]:
```
TfidfVectorizer(max_df=0.95, min_df=0.05)
```

In [64]:
```
token_tweet = tfidf_vect.fit_transform(tweet_df_lemma)
```

In [65]:
```
df_rf= pd.DataFrame(token_tweet.toarray(), columns=tfidf_vect.get_feature_names())
```

In [66]:
```
df_rf.shape
```

Out[66]:
```
(10358, 107)
```

In [67]:
```
from sklearn.model_selection import train_test_split

train_x, test_x,train_y, test_y = train_test_split(df_rf,y, test_size = 0.2, random_state = 5)
print(train_x.shape, test_x.shape)
```

```
(8286, 107) (2072, 107)
```

In [68]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
```

In [69]:
```
param_grid = {'n_estimators':[130,150,160,180,200],
              'max_features':[13,15,17,19]}

grid_rf = GridSearchCV(estimator= RandomForestClassifier(),
                       param_grid=param_grid,
                       cv = 10,
                       n_jobs=-1, verbose=True)

grid_rf.fit(train_x,train_y)
```

Out[69]:
```
Fitting 10 folds for each of 20 candidates, totalling 200 fits
GridSearchCV(cv=10, estimator=RandomForestClassifier(), n_jobs=-1,
             param_grid={'max_features': [13, 15, 17, 19],
                         'n_estimators': [130, 150, 160, 180, 200]},
             verbose=True)
```

In [70]:
```
grid_rf.best_score_
```

Out[70]:
```
0.8478183656462882
```

In [71]:
```
grid_rf.best_params_
```

Out[71]:
```
{'max_features': 15, 'n_estimators': 160}
```

In [72]:
```
grid_rf.best_estimator_
```

Out[72]:
```
RandomForestClassifier(max_features=15, n_estimators=160)
```

In [73]:
```
#Fit the model
rf_model = grid_rf.best_estimator_
rf_model.fit(train_x, train_y)
```

Out[73]:
```
RandomForestClassifier(max_features=15, n_estimators=160)
```

In [74]:
```
rf_train_predict = pd.DataFrame({'actual' : train_y,
                                 'predicted' : rf_model.predict(train_x)})
rf_train_predict.head()
```

Out[74]:

	actual	predicted
8233	0	0
5263	0	0
1756	0	0
2649	0	0
7023	0	1

In [75]:
```
rf_test_predict = pd.DataFrame({'actual' : test_y,
                                'predicted' : rf_model.predict(test_x)})
rf_test_predict.head()
```

Out[75]:

	actual	predicted
1988	0	0
442	0	0

	actual	predicted
5690	1	0
9844	0	0
9271	1	0

Model Evaulation

In [76]:
```
print('Accuracy Score for train dataset : ' , metrics.accuracy_score(rf_train_predict.actual, rf_train_predict.predicted))
print('Accuracy Score for test dataset : ' , metrics.accuracy_score(rf_test_predict.actual, rf_test_predict.predicted))
```

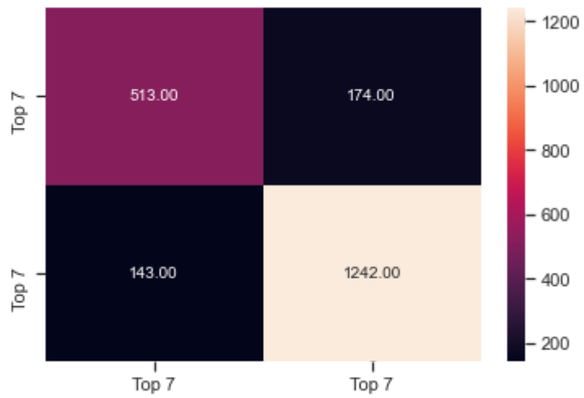
```
Accuracy Score for train dataset :  0.9482259232440261
Accuracy Score for test dataset :  0.847007722007722
```

In [77]:
```
print('ROC-AUC Score for train dataset : ' , metrics.roc_auc_score(rf_train_predict.actual, rf_train_predict.predicted))
print('ROC-AUC Score for validation dataset : ' , metrics.roc_auc_score(rf_test_predict.actual, rf_test_predict.predicted))
```

```
ROC-AUC Score for train dataset :  0.9417650203153465
ROC-AUC Score for validation dataset :  0.821737896678385
```

In [78]:
```
conn_cm_test = metrics.confusion_matrix(rf_test_predict.actual, rf_test_predict.predicted, [1,0])
sns.heatmap(conn_cm_test, fmt= '.2f', annot=True,  xticklabels=['Top 7', 'Top 7'], yticklabels=['Top 7', 'Top 7'])
```

Out[78]:
```
<AxesSubplot:>
```



In [79]:
```
print(metrics.classification_report(rf_test_predict.actual, rf_test_predict.predicted))
```

```
              precision    recall  f1-score   support

           0       0.88      0.90      0.89      1385
           1       0.78      0.75      0.76       687

    accuracy                           0.85      2072
   macro avg       0.83      0.82      0.83      2072
weighted avg       0.85      0.85      0.85      2072
```

In [80]:
```
indices = np.argsort(rf_model.feature_importances_)[::-1]
feature_rank = pd.DataFrame(columns = ['rank', 'feature', 'importance'])
for f in range(train_x.shape[1]):
    feature_rank.loc[f] = [f+1,
                           train_x.columns[indices[f]],
                           rf_model.feature_importances_[indices[f]]]
feature_rank.round(3)
```

Out[80]:

	rank	feature	importance
0	1	medium	0.088
1	2	joe	0.049
2	3	hillary	0.049
3	4	democrat	0.044
4	5	fake	0.043
...	...	...	...
102	103	know	0.003
103	104	because	0.003
104	105	one	0.002
105	106	trump	0.002
106	107	made	0.002

107 rows × 3 columns

In [81]:
```
feature_rank[:17]
```

Out[81]:

	rank	feature	importance
0	1	medium	0.087917
1	2	joe	0.048821
2	3	hillary	0.048796
3	4	democrat	0.043808
4	5	fake	0.043378
5	6	crooked	0.037967
6	7	news	0.037398
7	8	the	0.026681
8	9	cnn	0.023840
9	10	witch	0.018259
10	11	and	0.016607
11	12	of	0.015674
12	13	dems	0.015270
13	14	to	0.015261
14	15	hunt	0.015251
15	16	is	0.013660
16	17	he	0.012803

==========

#5a. -- Workings Detail... Ref Section Heading: [E] DEMO Tweet Generator

==========

DEMO : Tweet Generator using FastAI

Broadly, this is the summary of actions...

- First I will use a `different Trump-Tweet dataset` as training data input for generator
- Subsequently , I will use the `Trump-Insult dataset 2014-2021` to train the generator and see the diff

- `different Trump-Tweet dataset` is sourced from variety of sources, and includes a healthy mix of his tweets as well as about him (primarily from media outlets). My idea of this exercise is to demonstrate the impact of the insult dataset on the model.

P.S. Eventually these datasets are used for developing a BOT -- "Tweet BOT" which can tweet / target based on its persona analysis/ model training efficacy

In [111…

```
import pandas as pd
from tqdm import tqdm
import torch
from fastai.text.all import *
```

Pre-processing Texts

Preprocessing Function

In [112…

```
import re

def preprocess(text):
  o = re.sub(r'\\', ' ', text)
  k = re.sub(r'\n', ' ', o)
  l = re.sub(r'&', '', k)
  m = re.sub(r'RT ', '', l)
  n = re.sub(r'~', '', m)
  o = re.sub(r'#', '', n)
  p = re.sub(r'!+', '', o)
  q = re.sub(r'(http[s]?\S+)|(\w+\.[A-Za-z]{2,4}\S*)', 'link', p)
  r = re.sub(r'[*]', '', q)
  s = re.sub(r'[@]\w+', 'user', r)
  t = re.sub(r'[:|;]', '', s)
  u = re.sub(r'[\\x]\W+|\d', '', t)
  v = re.sub(r'[\\x]\W+|\d', '', u)
  w = re.sub(r'[\\x]\w+', '', v)
  x = re.sub(r'     ', ' ', w)
  y = re.sub(r'    ', ' ', x)
  z = re.sub(r'  ', ' ', y)
  ProcessedTweet = z.lower()
  return ProcessedTweet
```

Grabbing the texts:

In [113…

```
data = pd.read_csv("realdonaldtrump.csv")
data = data['content']
data.columns = ['text']
train = data

train.to_csv("./train.csv")
```

In [124…

```
data.head(5)
```

Out[124…

```
0                  be sure to tune in and watch donald trump on late night with david letterman as he presents the top ten list tonight
1    donald trump will be appearing on the view tomorrow morning to discuss celebrity apprentice and his new book think like a champion
2                                        donald trump reads top ten financial tips on late show with david letterman link - very funny
3                                                    new blog post celebrity apprentice finale and lessons learned along the way link
4                          "my persona will never be that of a wallflower - i'd rather build walls than cling to them" --donald j. trump
Name: content, dtype: object
```

Preprocessing the texts finally:

In [114…]
```
for i in tqdm(train.index):
    train.loc[i] = preprocess(train.loc[i])
```

```
100%|████████████████████████████████████████████████████████████████████████████| 43352/43352 [00:04<00:0
0, 9907.83it/s]
```

Grabbing the Dataloader:

In [115…]
```
train = pd.read_csv("./train.csv")
```

In [116…]
```
dls = TextDataLoaders.from_df(train, path="./", text_col='content', valid_pct=0.3, is_lm=True)
```

Model -- trained on `different Trump-Tweet dataset` ...

In [118…]
```
model = language_model_learner(dls, AWD_LSTM)
```

100.00% [105070592/105067061 00:03<00:00]

In [119…]
```
model.fit_one_cycle(5)
```

epoch	train_loss	valid_loss	time
0	5.119482	4.483339	25:30
1	4.346421	3.879691	48:47
2	4.088033	3.721125	25:37
3	3.990300	3.670313	26:11
4	3.985085	3.663576	25:41

It took a while ~2+ hours to train the model, which seems reasonable given the compute resources available for 43k rows in the training dataset for 5 cycle iterations. I had been watching movie simultaneously -- who you know how resources were used in parallel

Tweet Generating / Predicting using `different Trump-Tweet dataset`

Finally, "the moment of truth"...

for a pattern as "usa is greatest"

In [120…]
```
model.predict("usa is the greatest ", n_words=7)
```

Out[120…]
```
'usa is the greatest member and 1 to 5 women can'
```

well... let's now see what a counter-narrative were to say... a diff version?

In [121…]
```
model.predict("i think i am ", n_words=7)
```

Out[121…]
```
'i think i am going to win , but it is'
```

lets see fake news as input...

In [122…]
```
model.predict("you are fake news and you", n_words=7)
```

Out[122… 'you are fake news and you are LAT.MS correctly , i think'

In [123…
```
model.predict("i love ", n_words=15)
```

Out[123… 'i love that fact ! The term was always building for me , and i felt'

seems like Trump (according to this dataset) was like **dreamy... ?**

Let's see now if the model were to be trained on his 2014-2021 tweet dataset how it would spit out...

Now re-do the steps using FAST AI with `Trump-insult dataset 2014-2021`

In [127…
```
# intentionally taking Tweet and not insults, since I dont want to bias/ influence the model
# rather changing the tweet source from a mix of outsider + self to self alone
train_orig = df['tweet']
train_orig.head(3)
```

Out[127…
```
0    Can you believe this fool, Dr. Thomas Frieden of CDC, just stated, "anyone with fever should be asked if they have been in West Africa" DOPE
1    Can you believe this fool, Dr. Thomas Frieden of CDC, just stated, "anyone with fever should be asked if they have been in West Africa" DOPE
2                      Big time in U.S. today - MAKE AMERICA GREAT AGAIN! Politicians are all talk and no action - they can never bring us back.
Name: tweet, dtype: object
```

In [128…
```
train_orig.to_csv("./train_orig.csv")
```

In [129…
```
for i in tqdm(train_orig.index):
    train_orig.loc[i] = preprocess(train_orig.loc[i])
```

```
100%|██████████| 10358/10358 [00:02<00:00, 4594.71it/s]
```

In [130…
```
train_orig = pd.read_csv("./train_orig.csv")
```

In [132…
```
dls_orig = TextDataLoaders.from_df(train_orig, path="./", text_col='tweet', valid_pct=0.3, is_lm=True)
```

Model -- trained on `Trump-insult dataset 2014-2021`

In [134…
```
model_orig = language_model_learner(dls_orig, AWD_LSTM)
```

In [135…
```
model_orig.fit_one_cycle(5)
```

epoch	train_loss	valid_loss	time
0	5.112031	4.564421	08:20
1	4.637696	4.056053	08:11
2	4.311796	3.846397	08:14
3	4.140949	3.771640	08:12
4	4.078689	3.761346	08:09

In [ ]:
```
So this dataset is ~10k rows, so these 5 cycles complete in ~40 mins
```

Tweet Generating / Predicting using `Trump-insult dataset 2014-2021`

for the same pattern as "usa is greatest" *(as above)*

In [136…

```
model_orig.predict("usa is the greatest ", n_words=7)
```

Out[136…

```
'usa is the greatest man ever in America , together'
```

well... let's now see what a counter-narrative were to say... a diff version? *(as above)*

In [137…

```
model_orig.predict("i think i am ", n_words=7)
```

Out[137…

```
'i think i am fat , miners do not have in'
```

lets see fake news as input... *(as above)*

In [138…

```
model_orig.predict("you are fake news and you", n_words=7)
```

Out[138…

```
'you are fake news and you know what to do with them .'
```

In [139…

```
model_orig.predict("i love ", n_words=15)
```

Out[139…

```
'i love tour in the Wild , the play of Charlie Moore we see'
```

this time Trump doesn't sound "dreamy" rather **"ready to get back at it"..?**

When the model is trained on the dataset of Trump's own tweets, the generated tweet seem to match Trump's public image. With the different dataset, the tweets didn't exactly align or doesn't seem to be perceived the same way as intended. The difference in either datasets apart from the volume of rows, is the source of truth. Media tweets of trump are part of the dataset where the generated tweets seem "dreamy" unlike Trump

****

****