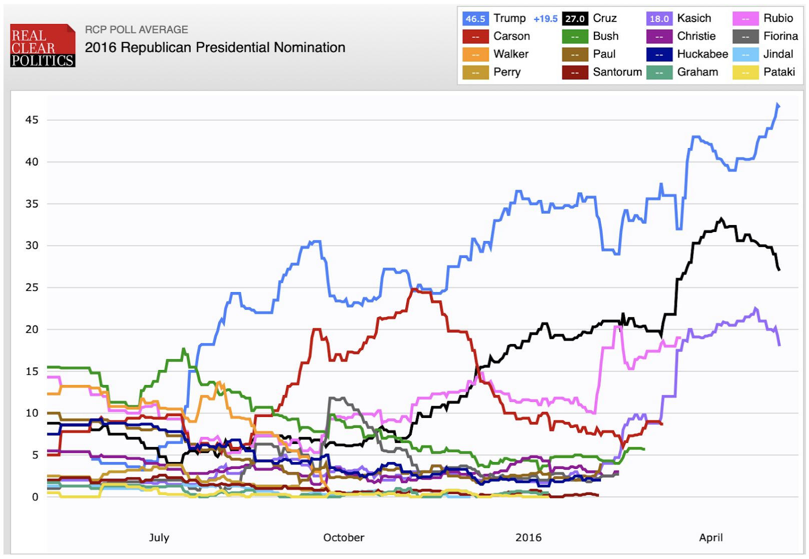
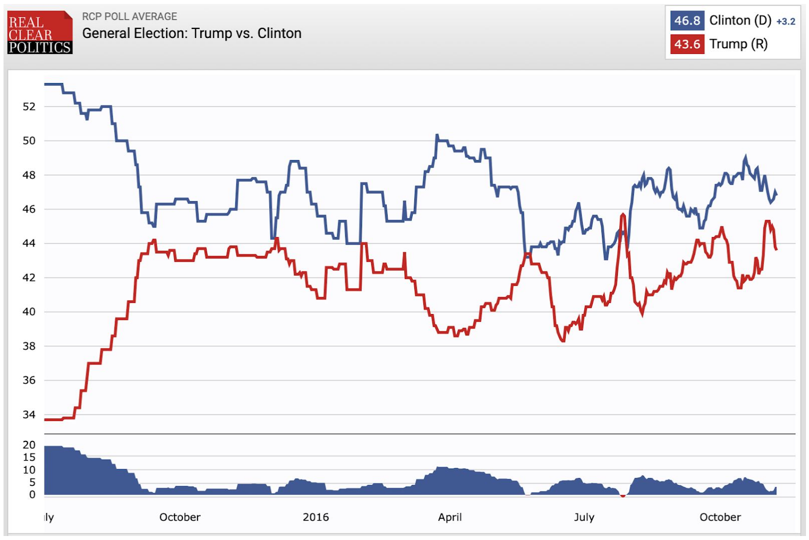
Classifying Tweets: A Presidential Case Study

"I am officially running for President of the United States. #MakeAmericaGreatAgain"

- @realdonaldtrump

On June 16th, 2015, at 3:57 PM, Donald J. Trump posted the above message to his Twitter account. Less than five weeks later, he was leading in the national polls to be the 2016 Republican Party Nominee for President of the United States of America. The following year, on July 19th, 2016, Trump was formally nominated by the delegates of the 2016 Republican National Convention. Finally, on November 8th of the same year, Trump defeated Hillary Clinton and became the 45th US President.

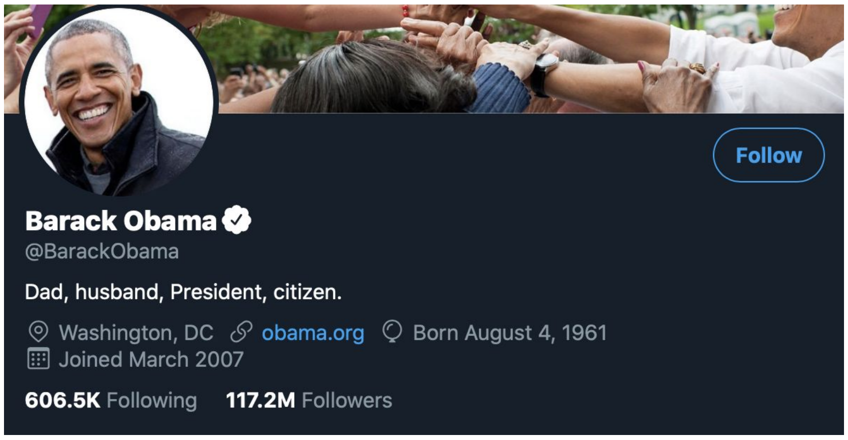




Plots courtesy of www.realclearpolitics.com

There are many different factors that led to the series of events described above. If you asked ten different people how Trump was able to pull off such an impressive victory, you might get ten different answers. One thing that everyone will agree on, though, is that Trump's ability to use social media to his advantage was a huge part of his strategy to gain popularity and help propel him to the highest office in the nation and arguably the most powerful position in the world.

A sound social media strategy is absolutely necessary to launch any successful business or political endeavor in today's world. Barack Obama, the 44th and previous president of the United States, was perhaps the first US president to truly take advantage of and leverage the power of social media. You'll now find him ranked #1 as the owner of the most followed Twitter account in the world, with over 117 million followers.



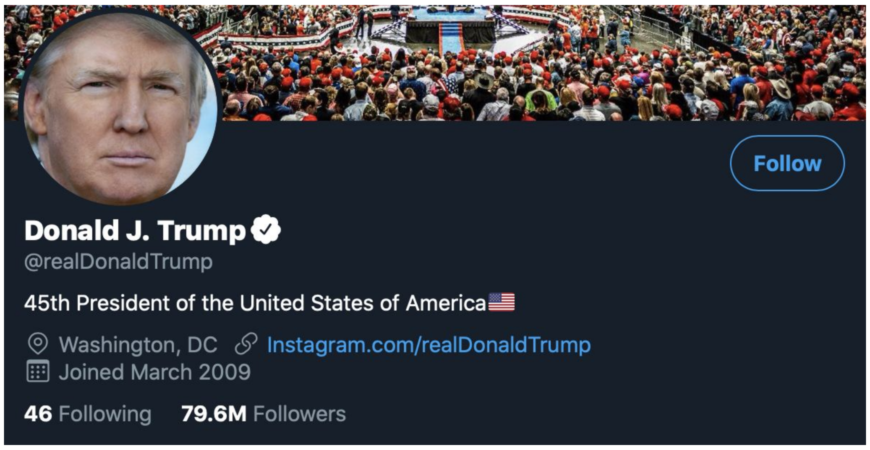
Barack Obama
@BarackObama

Dad, husband, President, citizen.

Washington, DC obama.org Born August 4, 1961
Joined March 2007

606.5K Following **117.2M** Followers

Twitter also happens to be president Trump's social media platform of choice. He finds himself ranked in the world's top ten most followed accounts as well, with over 79 million followers, sandwiched between TV host Ellen DeGeneres and pop musician Ariana Grande.



Donald J. Trump
@realDonaldTrump

45th President of the United States of America🇺🇸

Washington, DC Instagram.com/realDonaldTrump
Joined March 2009

46 Following **79.6M** Followers

What can we learn by studying the data created from a Twitter account? How and when does Trump use his account? Has becoming the President of the United States changed his tweeting habits? Is there a discernible difference between the tweets he wrote before and after becoming the president?

This project will attempt to answer these questions using the text and metadata generated by President Trump's Twitter account since its inception in March of 2009 by leveraging Natural Language Processing and Transfer Learning techniques. To do that, first we need to collect and organize the Twitter data.

Data Acquisition and Wrangling

the data collection process was smooth and painless thanks to the fine gentleman over at

www.trumptwitterarchive.com/archive

In the FAQ section of the website, I saw that the JSON files holding the data were being uploaded to his Github repository. Upon further inspection, I realized that most of the data was indeed there, but the most recent data from both 2019 and 2020 were missing. Rather than scraping his webpage or even worse, Trump's Twitter feed, I decided to do the next best thing: I found him on LinkedIn and reached out! Not only did Brendan respond quickly and patiently explain how I could use my browser's Web Inspector to download the JSONs directly from his website, but I also gained a new LinkedIn connection.

Now that the data was downloaded and stored locally, it was time to inspect the JSONs, turn them into a Pandas DataFrame, and continue engineering and extracting useful features and information for analyzing down the road. Each JSON held the Twitter data corresponding to a single year's worth of tweets.

Upon closer inspection, it became clear that the 2019 retweet metadata was corrupted. For some unknown reason, a significant interval of retweets were incorrectly classified as `False`. Since the text of each retweet starts with `'RT @'`, we can use the following code to correct the error:

```
df19['is_retweet'] = df19.apply(lambda row: True
                                 if row['text'].startswith('RT @')
                                 else False, axis=1)
```

Now we can concatenate all of the yearly DataFrames, sort the tweets into chronological order, and reset the index:

```
years = [df09, df10, df11, df12, df13, df14,
         df15, df16, df17, df18, df19, df20]

df = pd.concat(years)
df = df.sort_values('created_at').reset_index(drop=True)
```

Here is the column info on our preliminary DataFrame:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 47951 entries, 0 to 47950
Data columns (total 8 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   source                   47951 non-null  object
 1   id_str                   47951 non-null  int64
 2   text                     47951 non-null  object
 3   created_at               47951 non-null  datetime64[ns, UTC]
 4   retweet_count            47951 non-null  int64
 5   in_reply_to_user_id_str  3294 non-null   float64
 6   favorite_count           47951 non-null  int64
 7   is_retweet               47951 non-null  bool
dtypes: bool(1), datetime64[ns, UTC](1), float64(1), int64(3), object(2)
memory usage: 2.6+ MB
```

And here is the first row of the DataFrame, corresponding to Donald Trump's first ever tweet:

```
df.iloc[0]
```

```
source                                                           Twitter Web Client
id_str                                                                   1698308935
text                     Be sure to tune in and watch Donald Trump on Late Night with David L...
created_at                                                    2009-05-04 18:54:25+00:00
retweet_count                                                                       253
in_reply_to_user_id_str                                                             NaN
favorite_count                                                                      202
is_retweet                                                                        False
Name: 0, dtype: object
```

- The `'source'` column states what kind of device the tweet originated from.
- The `'id_str'` is simply a unique identifier for each tweet.
- The `'text'` column contains the text of the actual tweet as a string.
- The `'created_at'` column is a datetime object corresponding to the time of the tweet.
- The `'retweet_count'` tells how many times this particular tweet was shared by other users.
- The `'in_reply_to_user_id_str'` identifies the user this tweet was in response to.
- The `'favorite_count'` is the number of times this particular tweet was favorited.
- The `'is_retweet'` column is `True` if it's a retweet, and `False` otherwise.

The 'created_at' datetime has a lot of information packed in, so let's split off the date, time, and hour, into their own columns:

```
df['created_date'] = [d.date() for d in df['created_at']]
df['created_time'] = [d.time() for d in df['created_at']]
df['created_hour'] = pd.to_datetime(df['created_time'].astype(str)).dt.hour
```

We'll also pull some other useful information from the datetime object such as year, month, and day of the week.

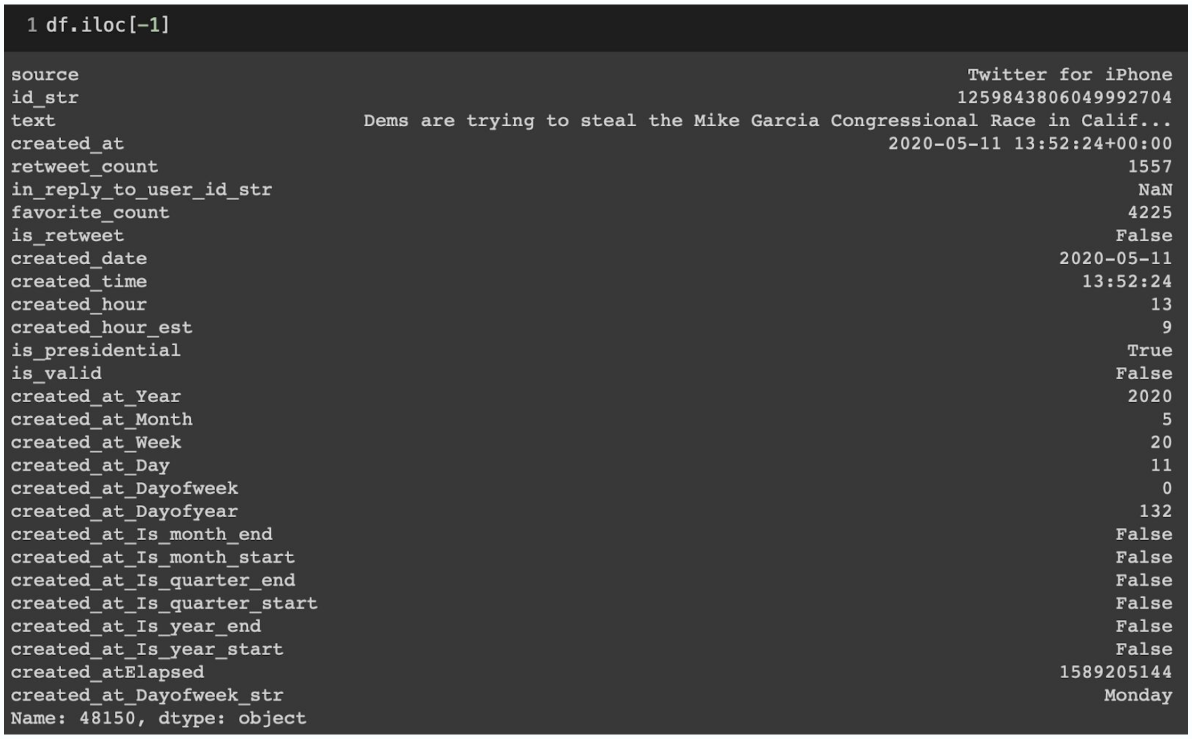
Since the time is being recorded in GMT, we should have a separate column for EST, since President Trump appears to spend most of his time between New York, Florida, and Washington D.C., all of which are on the East Coast:

```
def gmt_to_est(gmt):
  est = (gmt - 4) % 24
  return est

df['created_hour_est'] = df.apply(lambda row: gmt_to_est(row['created_hour']),
                                  axis=1)
```

Finally, since most of our analysis will be centered around a comparison of the tweets Trump has sent from both before and after becoming president, let's add a column `'is_presidental'` to return True or False depending on if it was sent while he was president or not:

```
df['is_presidential'] = df.apply(lambda row: True
                                 if row['created_date'] >= dt.date(2017,1,20)
                                 else False, axis=1)
```

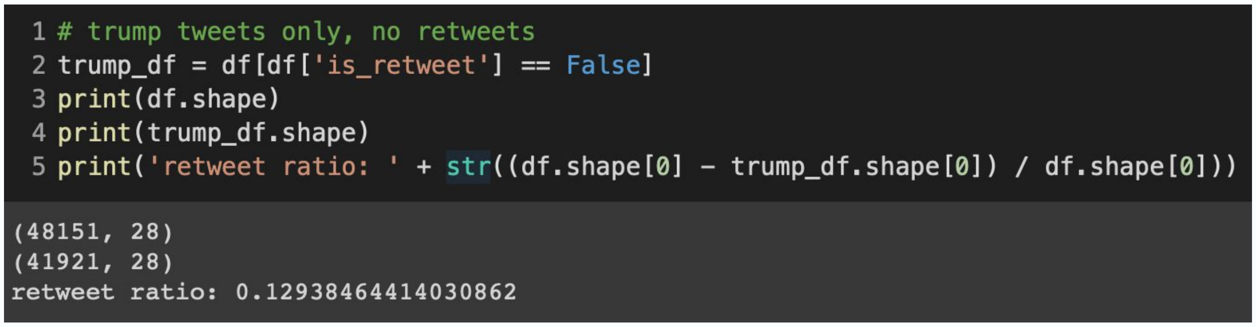
Let's take a look at the most recent tweet on file in the final DataFrame we have to work with:



```
df.iloc[-1]
```

```
source                                                                       Twitter for iPhone
id_str                                                                      1259843806049992704
text                           Dems are trying to steal the Mike Garcia Congressional Race in Calif...
created_at                                                            2020-05-11 13:52:24+00:00
retweet_count                                                                             1557
in_reply_to_user_id_str                                                                    NaN
favorite_count                                                                            4225
is_retweet                                                                               False
created_date                                                                        2020-05-11
created_time                                                                          13:52:24
created_hour                                                                                13
created_hour_est                                                                             9
is_presidential                                                                           True
is_valid                                                                                 False
created_at_Year                                                                           2020
created_at_Month                                                                             5
created_at_Week                                                                             20
created_at_Day                                                                              11
created_at_Dayofweek                                                                         0
created_at_Dayofyear                                                                       132
created_at_Is_month_end                                                                  False
created_at_Is_month_start                                                                False
created_at_Is_quarter_end                                                                False
created_at_Is_quarter_start                                                              False
created_at_Is_year_end                                                                   False
created_at_Is_year_start                                                                 False
created_atElapsed                                                                   1589205144
created_at_Dayofweek_str                                                                Monday
Name: 48150, dtype: object
```

Since we are mostly interested in just the original tweets that Trump creates himself, let's create a new DataFrame that ignores the retweets:



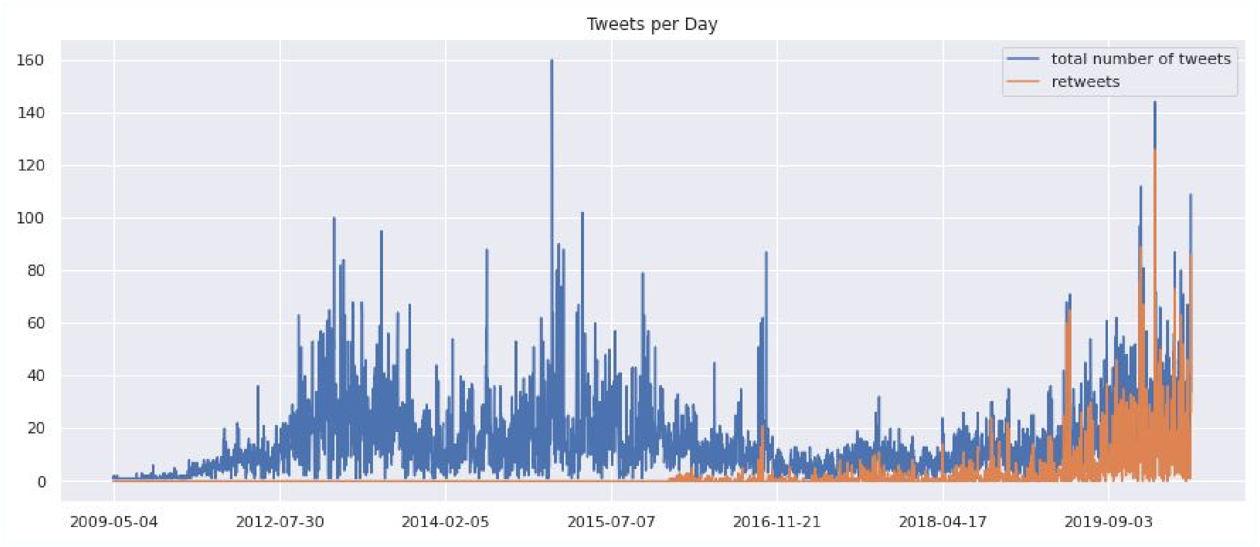
```python
# trump tweets only, no retweets
trump_df = df[df['is_retweet'] == False]
print(df.shape)
print(trump_df.shape)
print('retweet ratio: ' + str((df.shape[0] - trump_df.shape[0]) / df.shape[0]))
```

```
(48151, 28)
(41921, 28)
retweet ratio: 0.12938464414030862
```

We see that only about 12.9% of all of his tweets are retweets.

Exploratory Data Analysis

Let’s first take a look at the volume of president Trump’s tweets over time:



From this plot, we can learn quite a lot about Trump’s history using the platform. The first big rise and fall in volume occurred during his first presidential campaign in 2012. The next big peak, culminating in his largest single-day tweet output on January 5th, 2015, occurs during a season premier of his TV show, The Apprentice. Here are the first five tweets from that record-breaking day:

```
max_tweet_day['text'].head()
```

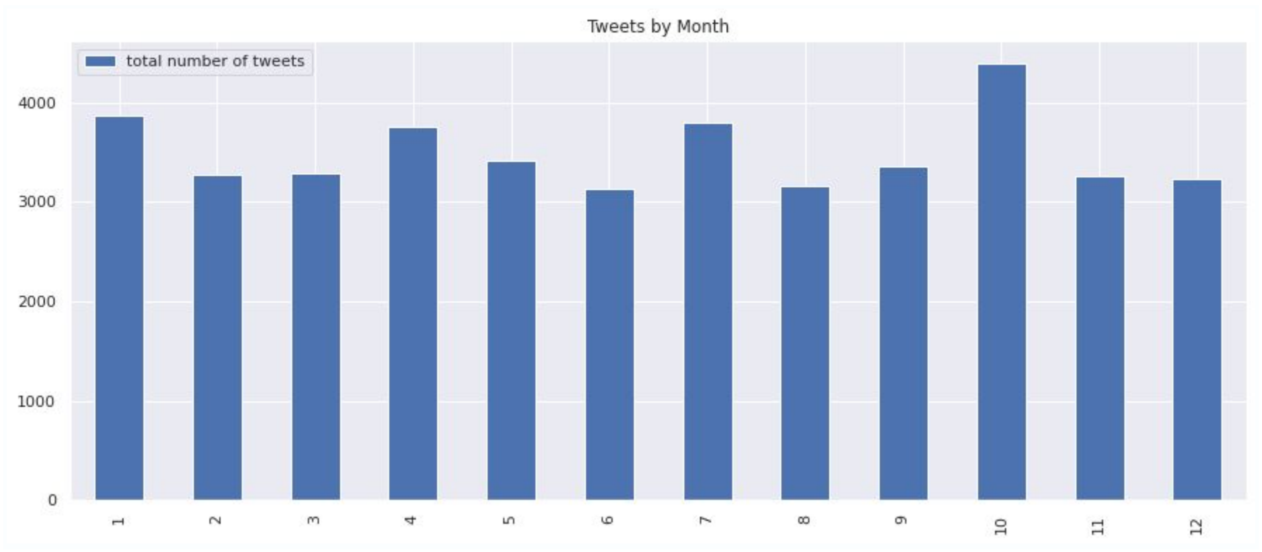
```
18560                          In less than 30 minutes- watch the season premiere of @ApprenticeNBC on NBC.
18561              This is going to be a special season - truly great characters and cast. You will soon see!
18562                  "@MLRatchford: @ApprenticeNBC @realDonaldTrump rooting for @RealGilbert. #TeamGilbert"
18563              "@fabiolasellsnj: @DonaldJTrumpJr @realDonaldTrump @ApprenticeNBC looking forward to this"
18564     "@Flowerchic462: Ok Tweeps. Who is going home first tonight?! #CelebApprentice @ApprenticeNBC @r...
Name: text, dtype: object
```

We can also check out his most favorited tweet of all time:

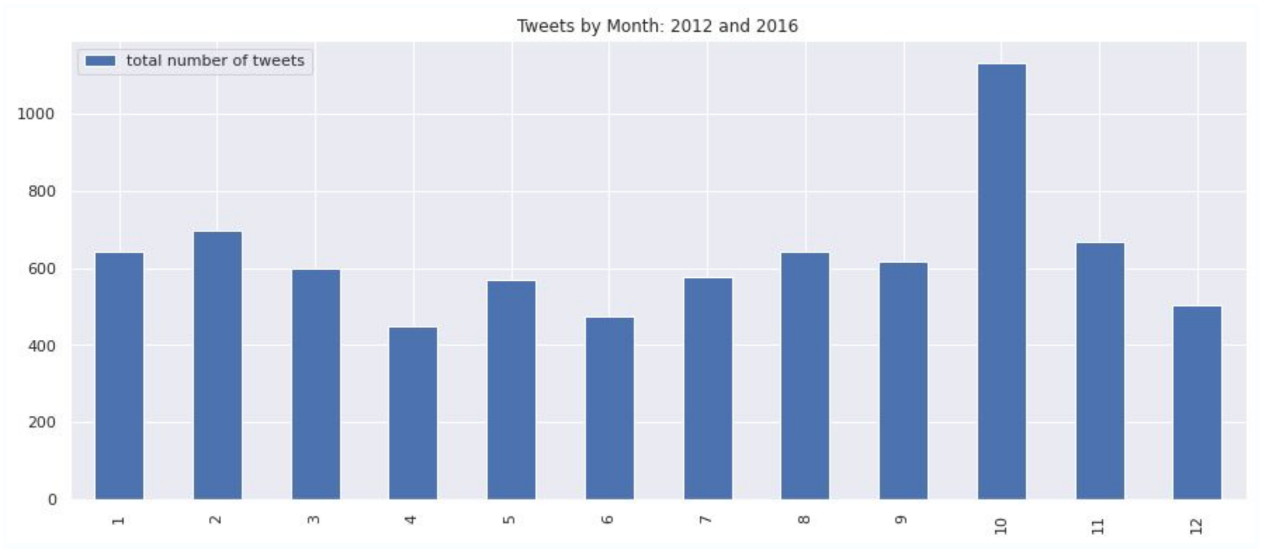
```
# Most favorited all time
trump_df.sort_values('favorite_count').iloc[-1]['text']
```

```
'A$AP Rocky released from prison and on his way home to the United States from
Sweden. It was a Rocky Week, get home ASAP A$AP!'
```

What time of year does Trump tweet the most?

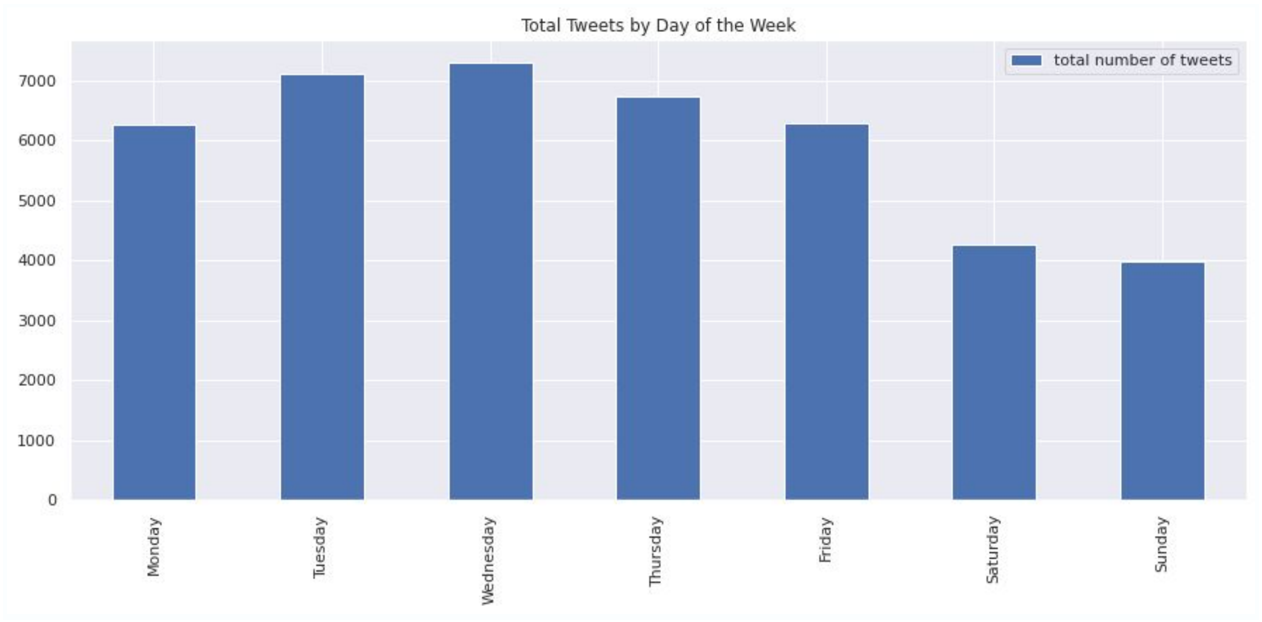


The most telling part of this plot is that it appears that Trump tweets the most often in October, known to many as the month right before most major US elections. Let’s look only at presidential election years where trump ran for office:



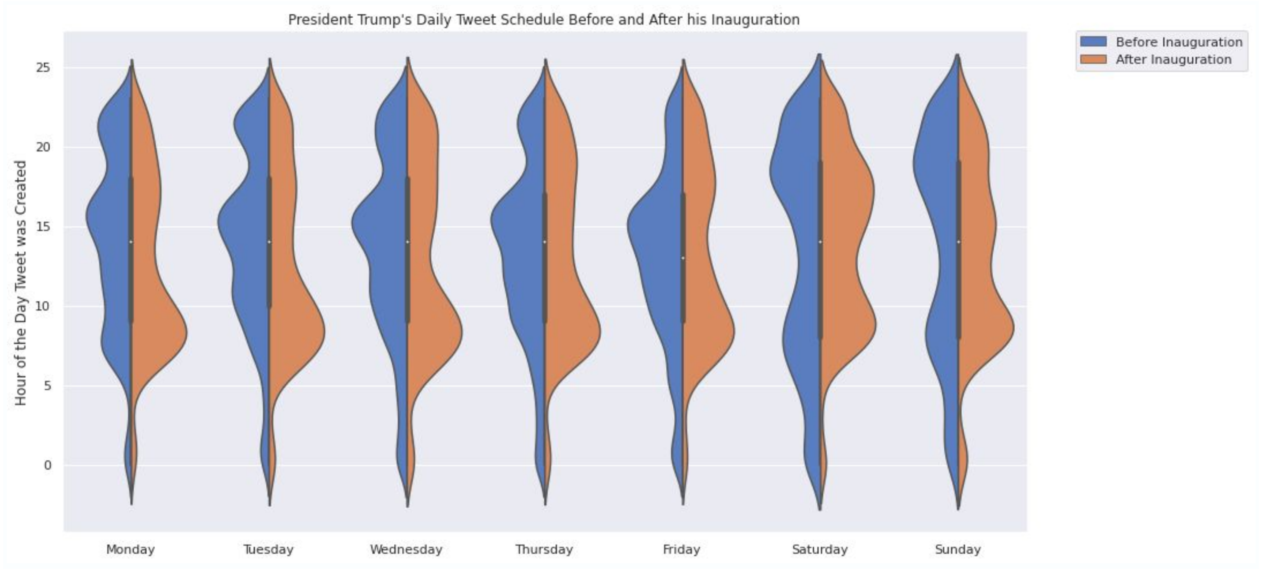
Sure enough, we see a much larger spike in October, confirming our suspicions that increase in tweet volume during October is most likely due to upcoming November elections.

What can we learn about President Trump’s daily and weekly schedule?



Trump appears to do most of his tweeting during the week and generates significantly less Twitter content on the weekends. This supports the notion that Trump approaches his use of Twitter as part of his daily job, as opposed to an extracurricular activity.

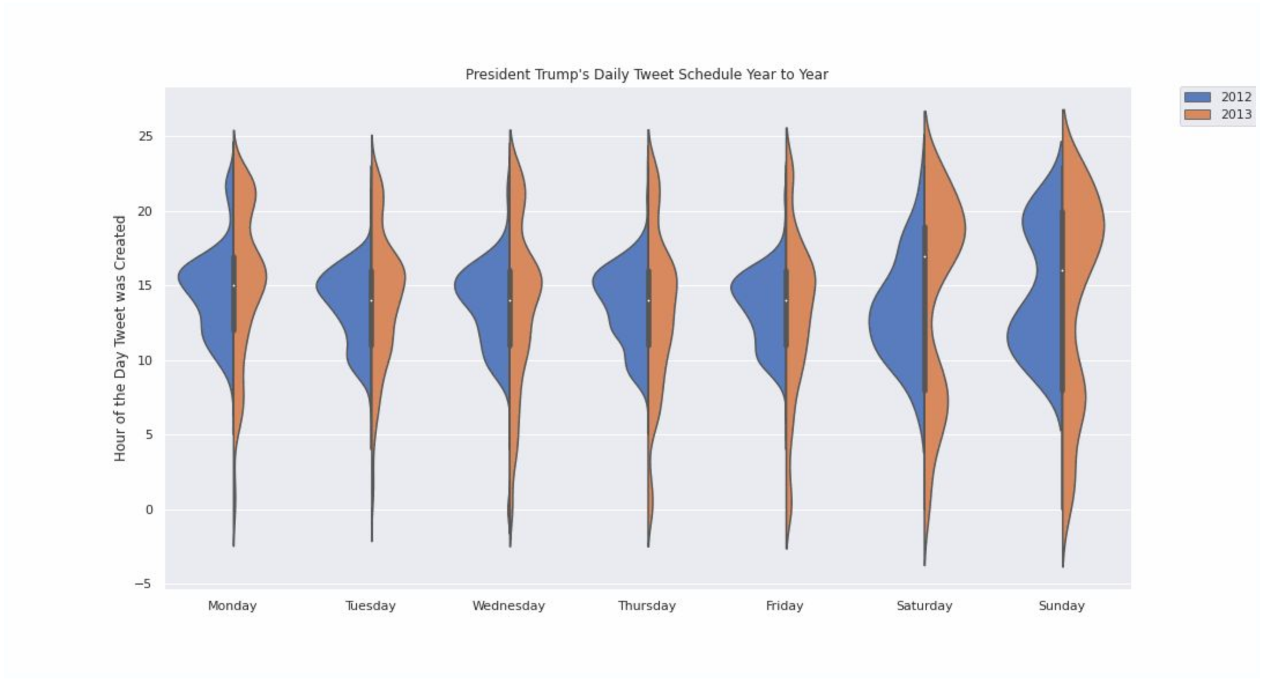
What about Trump’s daily Twitter schedule? Did becoming the president have any significant impact on what time throughout the day he tends to tweet the most?



It appears that prior to being elected, Trump was the most active on Twitter around 3 PM EST. Since being sworn in, though, his schedule has shifted significantly. As president, his use of Twitter appears to be heaviest in the mornings, between 7 and 9 AM EST.

Classifying Tweets: Is it Presidential?

In the previous plot, we saw that Trump's tweeting schedule had clearly shifted from the time before he was president, to the time after. But how can we be sure that this change occurred when he took on the job, and not at some other time? To better understand his schedule transition, let's look at the yearly schedule change from year to year with the following animated plot:



It's now more clear that his tweet schedule from year to year stays relatively uniform year to year up until the 2016, 2017 split. Not only was Trump inaugurated in the middle of this interval, but we see the most dramatic shift in his schedule during this time, as well. I think we can be more confident now in saying that his tweeting schedule was significantly impacted when he became president.

So what makes a tweet presidential? Simply put, a tweet is **presidential** if it was **tweeted while Trump was president.** That makes all of Trump's tweets after his inauguration date presidential, and every tweet before that date is not. We have already seen that Trump's presidential tweets follow a different daily schedule than the tweets sent out by Trump the civilian. But besides the time of day, are there other differences in tweets that are presidential and tweets that are not?

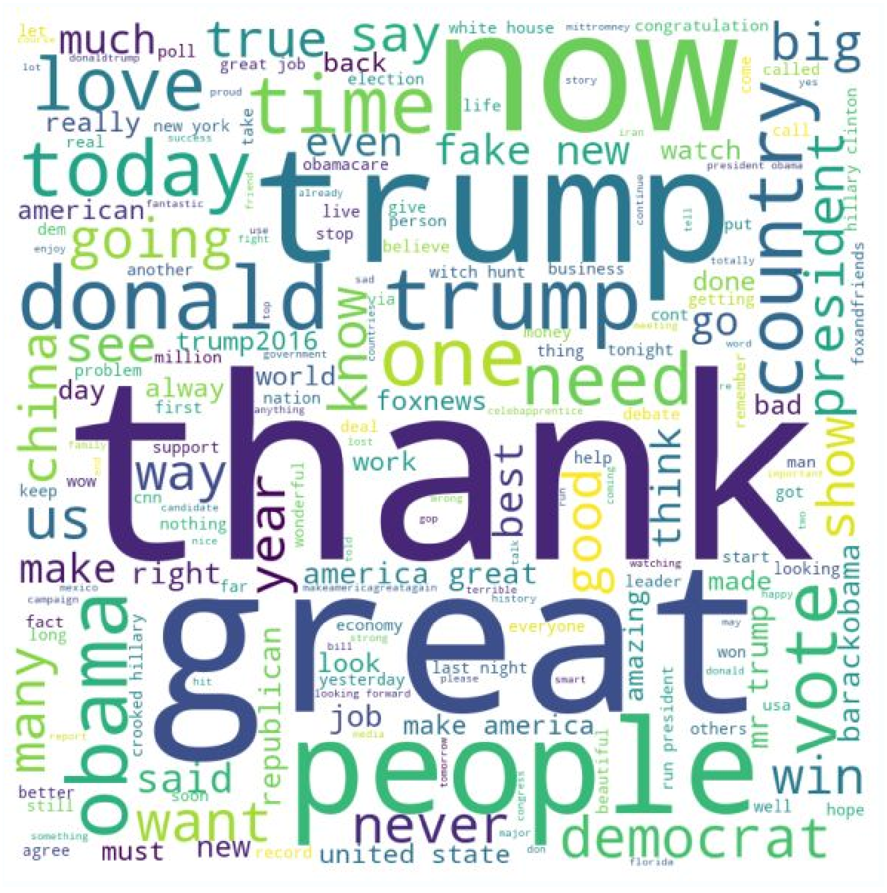
To find out, we'll take advantage of **fastai's** library and use **Transfer Learning** to train a **Neural Network** to classify tweets as being presidential or not by looking at the **text of the tweet only.** Are there enough qualitative differences between the two classes to distinguish the presidential tweets?

Fastai harnesses the power of Deep Learning and Transfer Learning to achieve state-of-the-art results with minimal training and smaller datasets. Transfer Learning refers to the idea that you can take a neural network that has already been trained on a large dataset and apply it to a different, yet related dataset by selectively training different parts of the network on the new data. For example, the fastai vision module uses well-established neural network architectures, like ResNet, and trains them on very large datasets, such as ImageNet, which currently has over 14 million images. The models are already capable of recognizing over a thousand classes of images. The early layers of the network are able to identify high-level abstract patterns that show up in most images, such as corners, edges, and basic shapes. The last few layers of the network are more specialized, and begin to recognize things like faces, cars, and animals. By retraining just the last layers of the network with a new set of data, and making only small adjustments to the rest of the weights in the network, we can ‘transfer’ the information learned previously to our new dataset and achieve very good results.

The fastai library also contains a text module for various Natural Language Processing tasks. We’ll use this module to build a classifier that can predict whether or not a Trump tweet was sent while he was president. We’ll use fastai’s pretrained **AWD-LSTM**. The model is a Long Short-Term Memory recurrent neural network that implements DropConnect on hidden-to-hidden weights. The model has been pre-trained on **Wikitext-103**, a collection of over 100 million tokens pulled from verified Good and Featured articles on Wikipedia. That means our model is already very good at understanding language and predicting next words; we only have to tweak and retrain parts of the model on our new dataset.

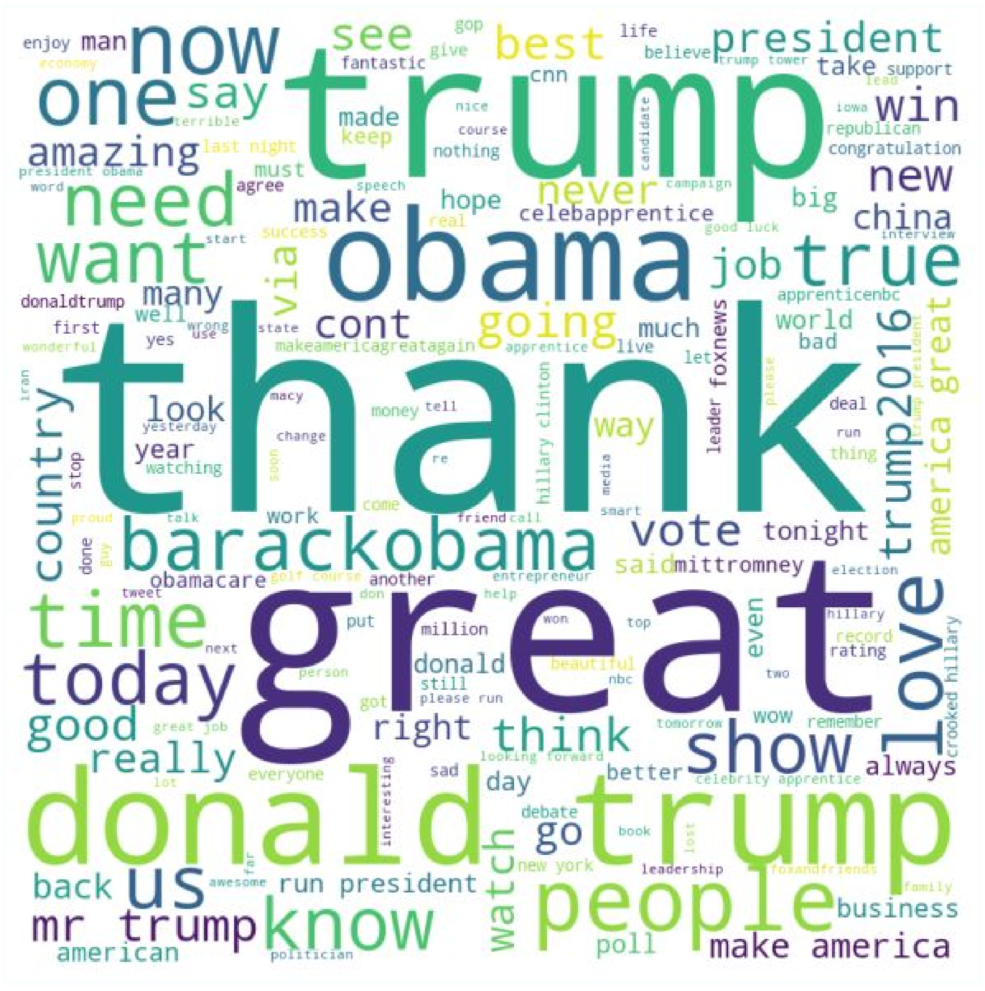
As before, we focus our attention on only the tweets created by Trump himself and ignore the retweets. We’ll use a wordcloud to briefly analyze the most used words across all of his tweets:

All Trump Tweets

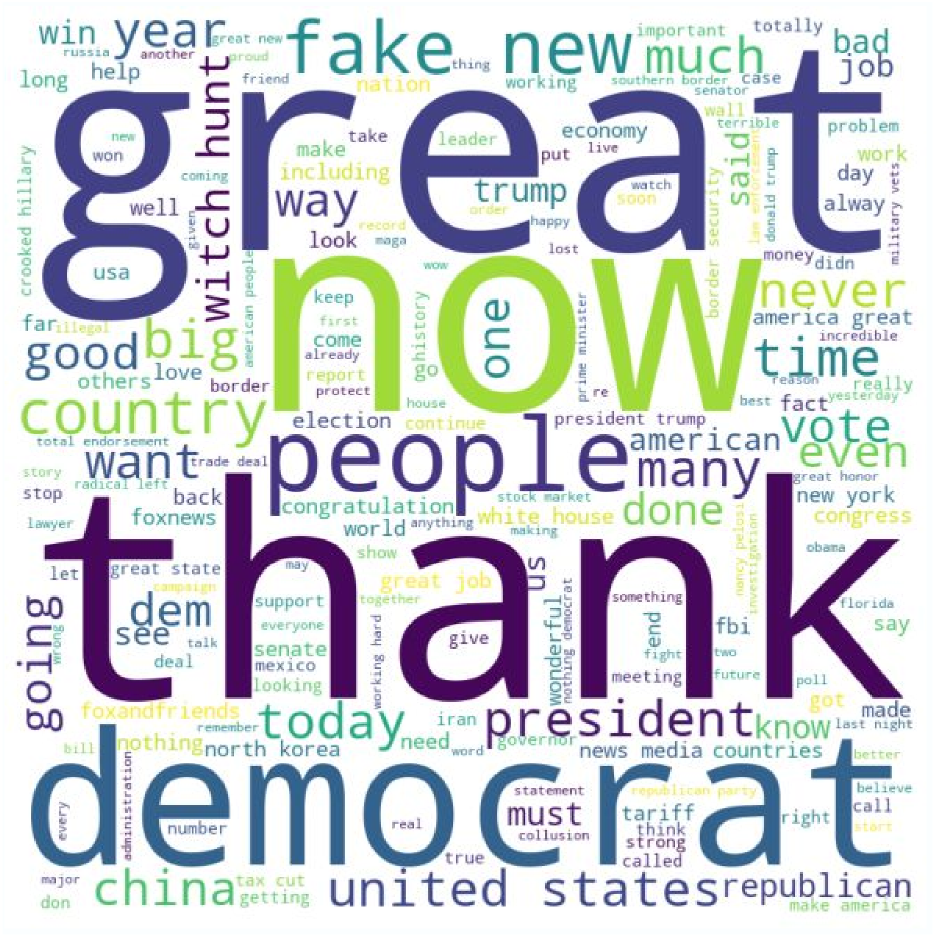


From here on, we'll refer to all of Trump's tweets that are not presidential as candidate tweets.

All Candidate Tweets

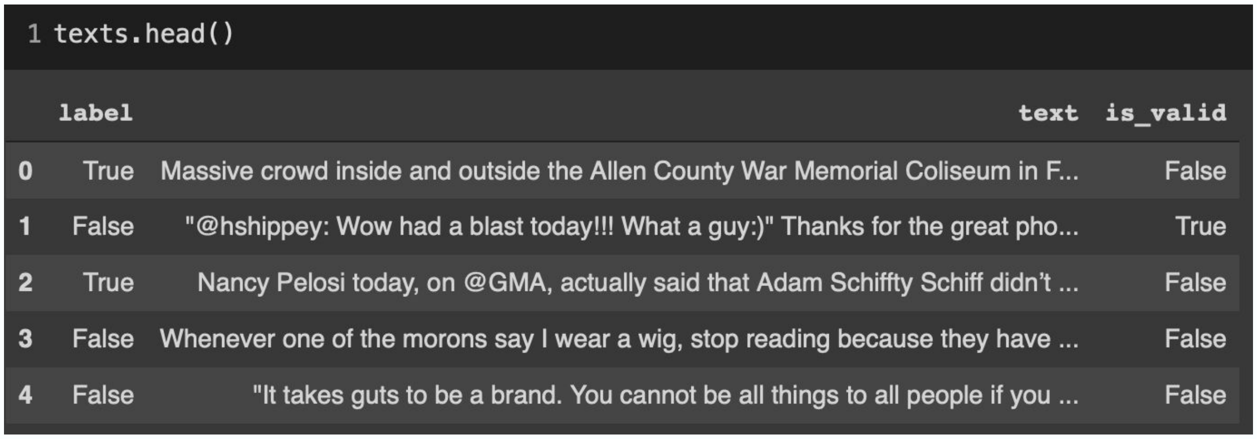


All Presidential Tweets



It's clear that many of the most popular words used show up in both the candidate tweets and the president tweets. Other words appear to be more prominent in one class over the other. Is there enough of a difference between the two classes to enable our classifier to make accurate predictions? Let's find out.

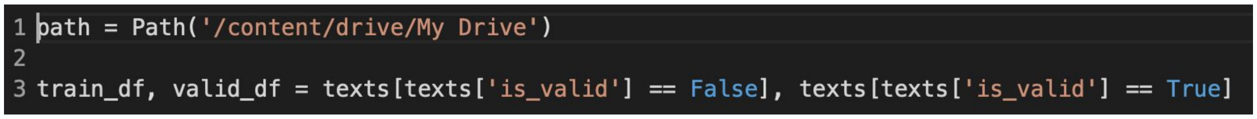
First, we need to prepare our dataset to make it fastai-friendly. We'll need a DataFrame with three columns: **label, text,** and **is_valid.** The label is our target variable, which in this case will be derived from the 'is_presidential' column. The 'is_valid' column will be to simplify the process of making a validation set, which will be 20% of all of the data, randomly chosen, and set aside for validation of the model. Next, we'll randomize the rows of the dataset, in order to eliminate any temporal data that might trickle into the model. We want the model to make it's prediction based only on the text of each tweet, and not on other factors such as it's relative location in the DataFrame. Here are the first five rows of our new DataFrame:



```python
texts.head()
```

	label	text	is_valid
0	True	Massive crowd inside and outside the Allen County War Memorial Coliseum in F...	False
1	False	"@hshippey: Wow had a blast today!!! What a guy:)" Thanks for the great pho...	True
2	True	Nancy Pelosi today, on @GMA, actually said that Adam Schiffty Schiff didn't ...	False
3	False	Whenever one of the morons say I wear a wig, stop reading because they have ...	False
4	False	"It takes guts to be a brand. You cannot be all things to all people if you ...	False

Next, we define the path where data and models will be stored throughout training.

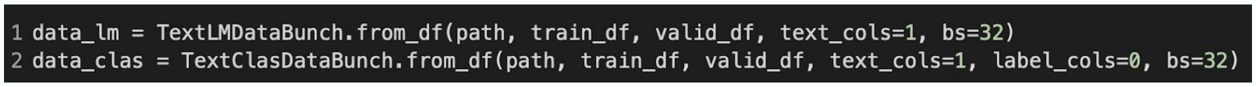


```python
path = Path('/content/drive/My Drive')

train_df, valid_df = texts[texts['is_valid'] == False], texts[texts['is_valid'] == True]
```

The process for building an NLP classifier with fastai is broken down into two main steps:

- Train a language model
- Use the language model to train the classifier

We therefore need to use the fastai data_block API to create two databunches: one for the language model, and one for the classifier.

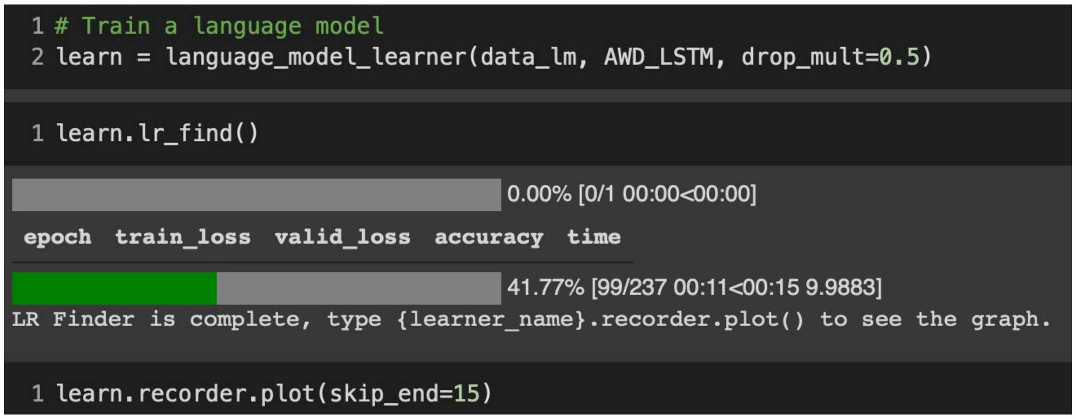


```python
data_lm = TextLMDataBunch.from_df(path, train_df, valid_df, text_cols=1, bs=32)
data_clas = TextClasDataBunch.from_df(path, train_df, valid_df, text_cols=1, label_cols=0, bs=32)
```

These databunches tokenize the tweets and separates punctuation into their own tokens, such is in the following example:

xxmaj carolina- thank you ! xxmaj heading to xxmaj johnstown , xxmaj pennsylvania now ! xxmaj get out on xxmaj november 8th … https : / / t.co / xxunk xxbos xxmaj newest xxmaj poll : xxmaj only 11 % in favor of starting ridiculous impeachment hearings . xxmaj well , let 's see : xxmaj we have the xxmaj best xxmaj economy in xxmaj history , the xxmaj best

We can now define the language model learner and find the optimal learning rate:

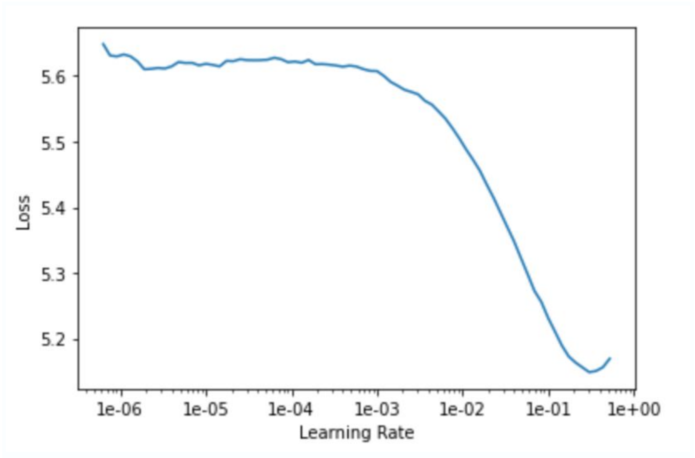


```
# Train a language model
learn = language_model_learner(data_lm, AWD_LSTM, drop_mult=0.5)
```

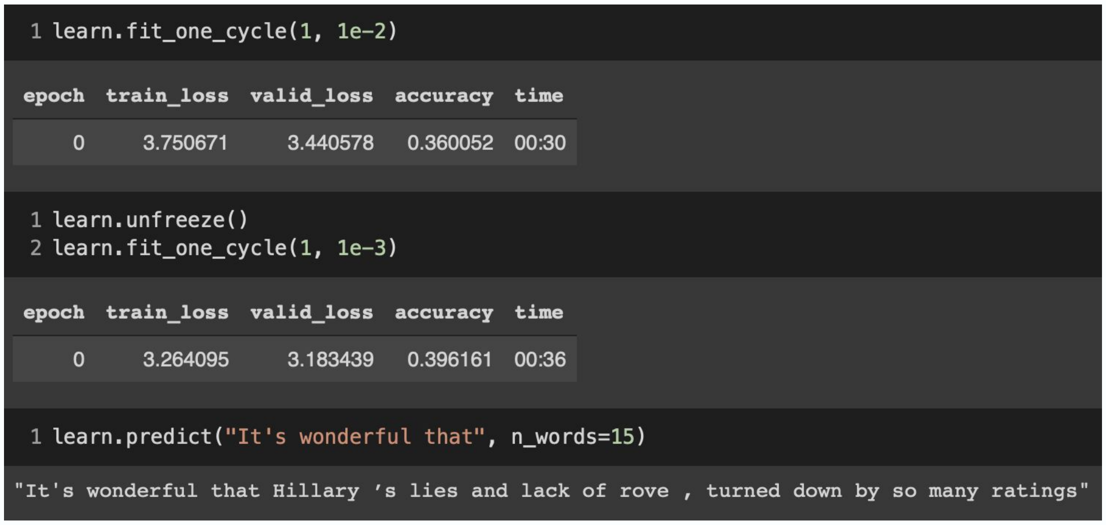
```
learn.lr_find()
```

```
0.00% [0/1 00:00<00:00]
epoch  train_loss  valid_loss  accuracy  time

41.77% [99/237 00:11<00:15 9.9883]
LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.
```

```
learn.recorder.plot(skip_end=15)
```



We choose a learning rate corresponding to when the Loss begins to fall rapidly. This occurs at approximately 1e-02. We'll use this learning rate to first fit the last layer of the model to our data, and then we'll unfreeze the model in order to fit all of the layers with a learning rate ten times smaller, in order to keep the weights from changing too much. The language model's job is to predict the next word in a string of words, and we can test it out at this point.



```
learn.fit_one_cycle(1, 1e-2)
```

```
epoch  train_loss  valid_loss  accuracy  time
    0    3.750671    3.440578  0.360052  00:30
```

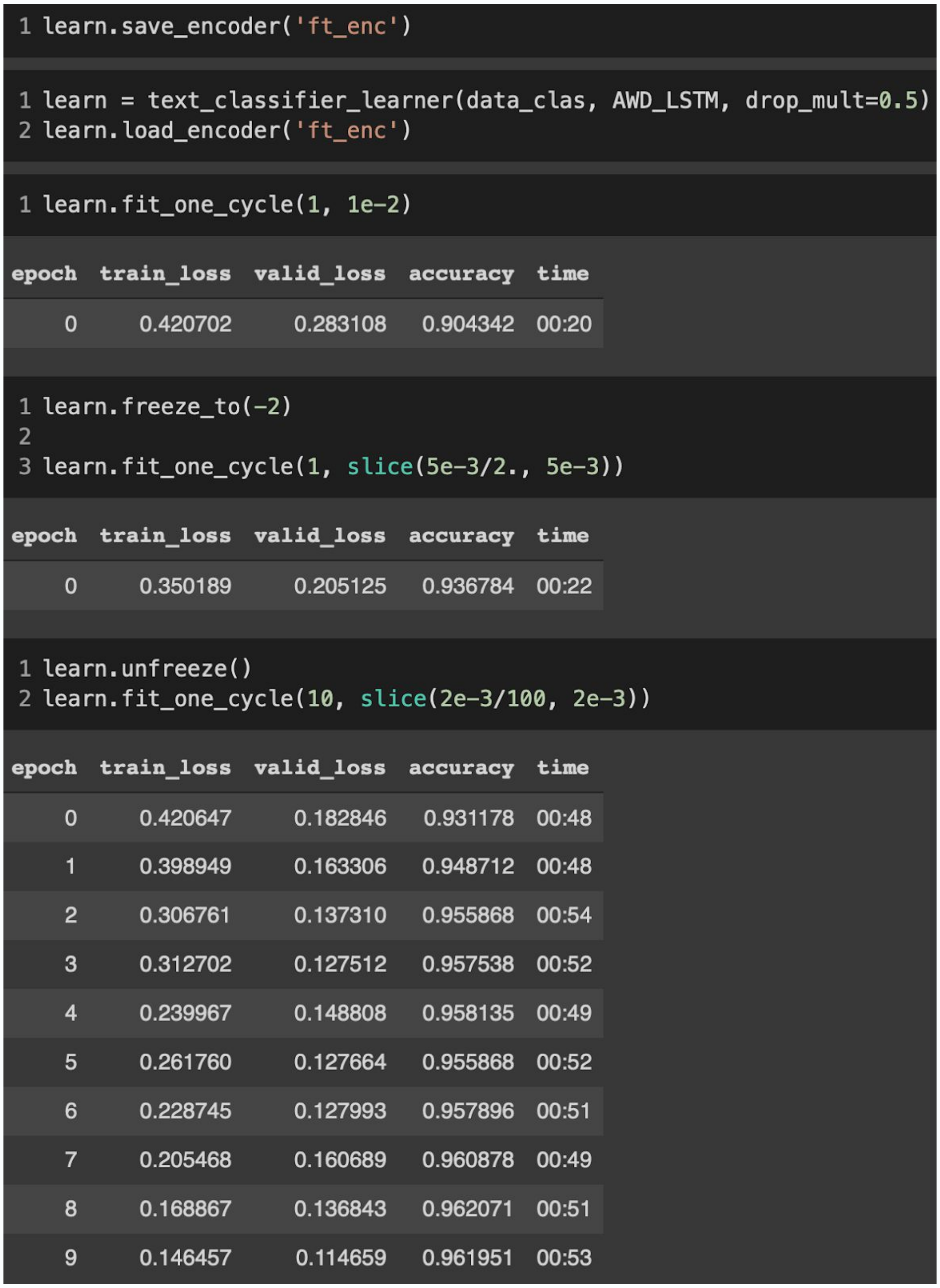
```
learn.unfreeze()
learn.fit_one_cycle(1, 1e-3)
```

```
epoch  train_loss  valid_loss  accuracy  time
    0    3.264095    3.183439  0.396161  00:36
```

```
learn.predict("It's wonderful that", n_words=15)
```

```
"It's wonderful that Hillary 's lies and lack of rove , turned down by so many ratings"
```

Next up is the classifier. We'll save the language model encoder as `'ft_enc'`, define the classifier, load the encoder, fit one cycle on the last layer, and finally unfreeze and fit on all the layers. In this last phase of training, we use a variable learning rate in the form of a slice. This will slowly increase the learning rate as the model trains.



```python
learn.save_encoder('ft_enc')
```

```python
learn = text_classifier_learner(data_clas, AWD_LSTM, drop_mult=0.5)
learn.load_encoder('ft_enc')
```

```python
learn.fit_one_cycle(1, 1e-2)
```

```
epoch  train_loss  valid_loss  accuracy  time
    0    0.420702    0.283108  0.904342  00:20
```

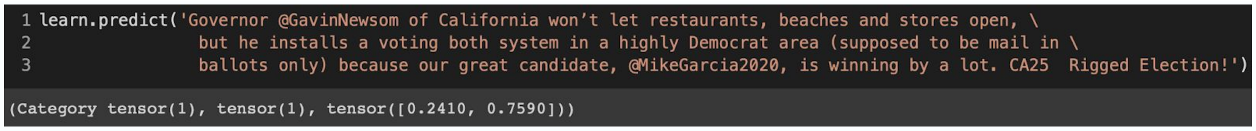
```python
learn.freeze_to(-2)

learn.fit_one_cycle(1, slice(5e-3/2., 5e-3))
```

```
epoch  train_loss  valid_loss  accuracy  time
    0    0.350189    0.205125  0.936784  00:22
```

```python
learn.unfreeze()
learn.fit_one_cycle(10, slice(2e-3/100, 2e-3))
```

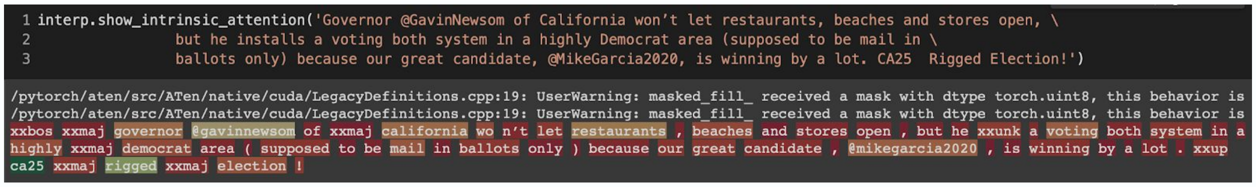
```
epoch  train_loss  valid_loss  accuracy  time
    0    0.420647    0.182846  0.931178  00:48
    1    0.398949    0.163306  0.948712  00:48
    2    0.306761    0.137310  0.955868  00:54
    3    0.312702    0.127512  0.957538  00:52
    4    0.239967    0.148808  0.958135  00:49
    5    0.261760    0.127664  0.955868  00:52
    6    0.228745    0.127993  0.957896  00:51
    7    0.205468    0.160689  0.960878  00:49
    8    0.168867    0.136843  0.962071  00:51
    9    0.146457    0.114659  0.961951  00:53
```

By the time training has finished, we’ve managed to build a classifier that can accurately predict a presidential tweet **96%** of the time. Let’s run the classifier on a recent tweet that was neither part of the training nor validation datasets:



```
learn.predict('Governor @GavinNewsom of California won’t let restaurants, beaches and stores open, \
               but he installs a voting both system in a highly Democrat area (supposed to be mail in \
               ballots only) because our great candidate, @MikeGarcia2020, is winning by a lot. CA25  Rigged Election!')

(Category tensor(1), tensor(1), tensor([0.2410, 0.7590]))
```

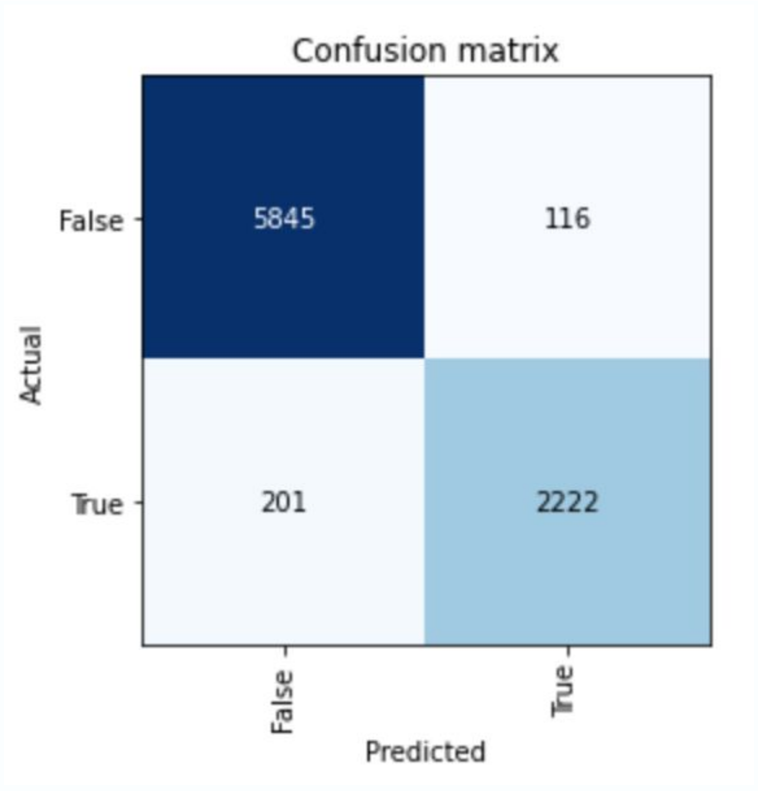
The classifier is predicting that this is a presidential tweet (correct!) with a confidence level of 75.9%. We can also look at the intrinsic attention given to each word:



```
interp.show_intrinsic_attention('Governor @GavinNewsom of California won’t let restaurants, beaches and stores open, \
               but he installs a voting both system in a highly Democrat area (supposed to be mail in \
               ballots only) because our great candidate, @MikeGarcia2020, is winning by a lot. CA25  Rigged Election!')

/pytorch/aten/src/ATen/native/cuda/LegacyDefinitions.cpp:19: UserWarning: masked_fill_ received a mask with dtype torch.uint8, this behavior is
/pytorch/aten/src/ATen/native/cuda/LegacyDefinitions.cpp:19: UserWarning: masked_fill_ received a mask with dtype torch.uint8, this behavior is
xxbos xxmaj governor @gavinnewsom of xxmaj california wo n’t let restaurants , beaches and stores open , but he xxunk a voting both system in a
highly xxmaj democrat area ( supposed to be mail in ballots only ) because our great candidate , @mikegarcia2020 , is winning by a lot . xxup
ca25 xxmaj rigged xxmaj election !
```

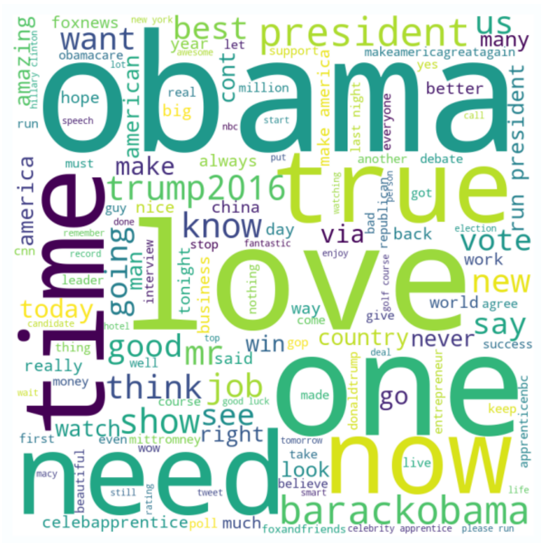
The most important word here is `'ca25'`, dark green in the lower left, which refers to California’s 25th district and the ongoing special election taking place. Perhaps Trump has only ever mentioned this district in reference to this election, so it makes for a good indicator that this is probably a presidential tweet.

Let’s now take a look at the confusion matrix:

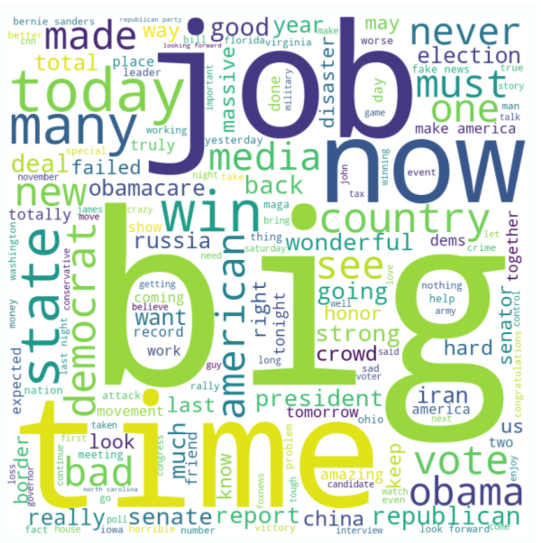


We can also look at the Confusion Matrix wordclouds to see which words stood out the most in the tweets that were classified both correctly and incorrectly ('thank', 'thanks', 'great', 'people', and 'trump' were ignored this round, since they showed up a large amount in each cloud):

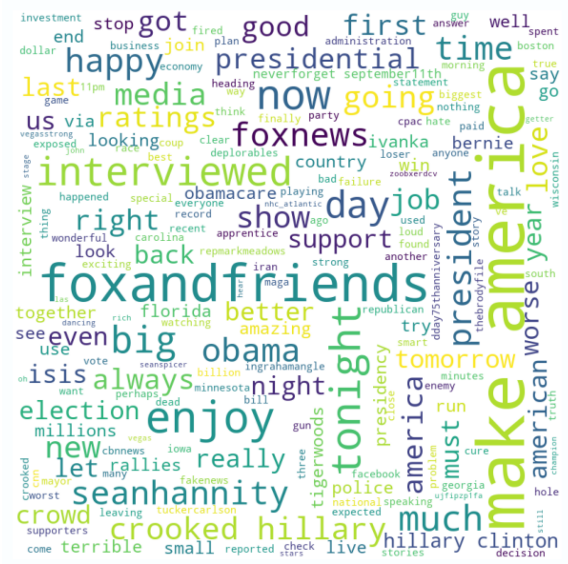
False Negatives



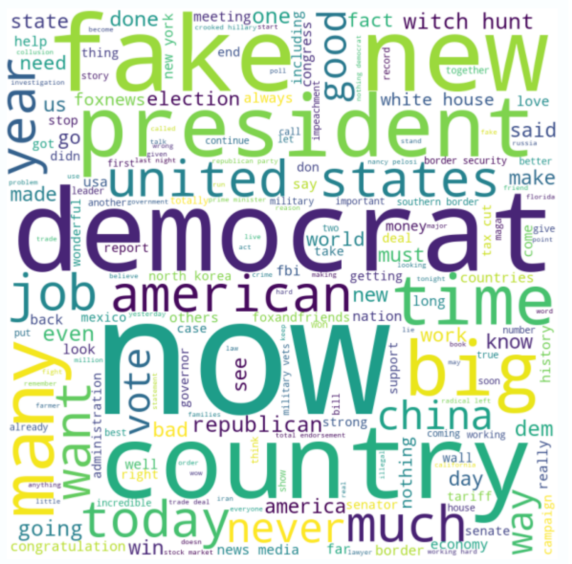
False Positives



True Negatives



True Positives



The candidate tweets that were predicted to be presidential (False Positives) seem to contain a handful of words that appear to the classifier as being very presidential in nature: Job, Now, Big, and Time. Interestingly, though, the presidential tweets that were predicted to be candidate tweets (True Negatives) don't really have any stand-out words.

Conclusion

Through a combination of data wrangling, exploratory data analysis, Deep Learning, and Transfer Learning, we've been able to gain useful insights and get a better picture of the nature of President Trump's Twitter output. We saw a clear shift in Trump's tweeting schedule occur when he became president, which may indicate an overall shift in the content of the tweets as well. The recurrent neural network we trained on his tweets resulted in a very accurate presidential tweet predictor that, most likely, outperforms an average American's attempt at the same classification problem (this would need to be tested.) Then, by analyzing the confusion matrix and corresponding word clouds, we were able to determine which words were showing up the most in the accurate predictions, as well as the predictions that turned out wrong.

The insights gained in this study showcase the power of fastai's Deep Learning modules. With a small dataset and a pretrained RNN, a very accurate classification model was developed with minimal training time and resources.