# Project 1: Trump, Twitter, and Text

In this project, we will work with the Twitter API in order to analyze Donald Trump's tweets.

**The project is due 11:59pm Sunday, October 20**

If you find yourself getting frustrated or stuck on one problem for too long, we suggest coming into office hours and working with friends in the class.

In [82]:

```
# Run this cell to set up your notebook
import csv
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import zipfile
import json

# Ensure that Pandas shows at least 280 characters in columns, so we can see full tweets
pd.set_option('max_colwidth', 280)

%matplotlib inline
plt.style.use('fivethirtyeight')
import seaborn as sns
sns.set()
sns.set_context("talk")
import re
```

## Getting the data

The starting point and a key aspect of any data science project is getting the data. To get Twitter data, Twitter conveniently provides a developer API using which we can scrape data. More on that will follow in the coming discussions!

For now, we've made life easier for you by providing the data.

Start by running the following cells, which will download and then load Donald Trump's most recent tweets.

In [83]:

```
# Download the dataset
from utils import fetch_and_cache
data_url = 'https://cims.nyu.edu/~policast/recent_tweets.json'
file_name = 'realdonaldtrump_recent_tweets.json'

dest_path = fetch_and_cache(data_url=data_url, file=file_name)
print(f'Located at {dest_path}')
```

```
Using version already downloaded: Mon Oct  7 20:46:23 2019
MD5 hash of file: 216176fb098cd5d6b40b373b98bd3e6d
Located at data/realdonaldtrump_recent_tweets.json
```

In [84]:

```
def load_tweets(path):
    """Loads tweets that have previously been saved.

    Calling load_tweets(path) after save_tweets(tweets, path)
    will produce the same list of tweets.

    Args:
        path (str): The place where the tweets were be saved.

    Returns:
        list: A list of Dictionary objects, each representing one tweet."""

    with open(path, "rb") as f:
        import json
        return json.load(f)
```

In [85]:

```
trump_tweets = load_tweets(dest_path)
```

If everything is working correctly correctly this should load roughly the last 3000 tweets by `realdonaldtrump` .

In [86]:

```
assert 2000 <= len(trump_tweets) <= 4000
```

If the assert statement above works, then continue on to question 2b.

## Question 1

We are limited to how many tweets we can download. In what month is the oldest tweet from Trump?

In [87]:

```
# Enter the number of the month of the oldest tweet (e.g. 1 for January)
### BEGIN SOLUTION
oldest_month = pd.to_datetime(pd.Series([temp['created_at'] for temp in trump_
tweets])).min().month
oldest_month
#TODO
### END SOLUTION
```

Out[87]:

```
10
```

In [88]:

```
### BEGIN HIDDEN TESTS
assert oldest_month > 9
assert oldest_month < 12
### END HIDDEN TESTS
```

**IMPORTANT! PLEASE READ**

What if we want to access Donald Trump's old tweets?
Unfortunately, you cannot download old tweets using the public Twitter APIs. Fortunately, we have a snapshot of earlier tweets of Donald Trump that we can combine with the newer data that you downloaded

We will again use the `fetch_and_cache` utility to download the dataset.

In [89]:

```
# Download the dataset
from utils import fetch_and_cache
data_url = 'https://cims.nyu.edu/~policast/old_trump_tweets.json.zip'
file_name = 'old_trump_tweets.json.zip'

dest_path = fetch_and_cache(data_url=data_url, file=file_name)
print(f'Located at {dest_path}')
```

```
Using version already downloaded: Mon Oct  7 20:46:23 2019
MD5 hash of file: b6e33874de91d1a40207cdf9f9b51a09
Located at data/old_trump_tweets.json.zip
```

Finally, we we will load the tweets directly from the compressed file without decompressing it first.

In [90]:

```
my_zip = zipfile.ZipFile(dest_path, 'r')
with my_zip.open("old_trump_tweets.json", "r") as f:
    old_trump_tweets = json.load(f)
```

This data is formatted identically to the recent tweets we just downloaded:

```
In [91]: print(old_trump_tweets[0])
```

```
{'created_at': 'Wed Oct 12 14:00:48 +0000 2016', 'id': 786204978629185536, 'i
d_str': '786204978629185536', 'text': 'PAY TO PLAY POLITICS. \n#CrookedHillar
y https://t.co/wjsl8ITVvk', 'truncated': False, 'entities': {'hashtags': [{'t
ext': 'CrookedHillary', 'indices': [23, 38]}], 'symbols': [], 'user_mention
s': [], 'urls': [], 'media': [{'id': 786204885318561792, 'id_str': '786204885
318561792', 'indices': [39, 62], 'media_url': 'http://pbs.twimg.com/ext_tw_vi
deo_thumb/786204885318561792/pu/img/XqMoixLm83FzkAbn.jpg', 'media_url_https':
'https://pbs.twimg.com/ext_tw_video_thumb/786204885318561792/pu/img/XqMoixLm8
3FzkAbn.jpg', 'url': 'https://t.co/wjsl8ITVvk', 'display_url': 'pic.twitter.c
om/wjsl8ITVvk', 'expanded_url': 'https://twitter.com/realDonaldTrump/status/7
86204978629185536/video/1', 'type': 'photo', 'sizes': {'thumb': {'w': 150,
'h': 150, 'resize': 'crop'}, 'medium': {'w': 600, 'h': 338, 'resize': 'fit'},
'small': {'w': 340, 'h': 191, 'resize': 'fit'}, 'large': {'w': 1024, 'h': 57
6, 'resize': 'fit'}}}]}, 'extended_entities': {'media': [{'id': 7862048853185
61792, 'id_str': '786204885318561792', 'indices': [39, 62], 'media_url': 'htt
p://pbs.twimg.com/ext_tw_video_thumb/786204885318561792/pu/img/XqMoixLm83FzkA
bn.jpg', 'media_url_https': 'https://pbs.twimg.com/ext_tw_video_thumb/7862048
85318561792/pu/img/XqMoixLm83FzkAbn.jpg', 'url': 'https://t.co/wjsl8ITVvk',
'display_url': 'pic.twitter.com/wjsl8ITVvk', 'expanded_url': 'https://twitte
r.com/realDonaldTrump/status/786204978629185536/video/1', 'type': 'video', 's
izes': {'thumb': {'w': 150, 'h': 150, 'resize': 'crop'}, 'medium': {'w': 600,
'h': 338, 'resize': 'fit'}, 'small': {'w': 340, 'h': 191, 'resize': 'fit'},
'large': {'w': 1024, 'h': 576, 'resize': 'fit'}}, 'video_info': {'aspect_rati
o': [16, 9], 'duration_millis': 30106, 'variants': [{'bitrate': 832000, 'cont
ent_type': 'video/mp4', 'url': 'https://video.twimg.com/ext_tw_video/78620488
5318561792/pu/vid/640x360/6vt24D3ZQSvYuDqe.mp4'}, {'bitrate': 2176000, 'conte
nt_type': 'video/mp4', 'url': 'https://video.twimg.com/ext_tw_video/786204885
318561792/pu/vid/1280x720/rSbgQdvR9TPIlRWr.mp4'}, {'bitrate': 320000, 'conten
t_type': 'video/mp4', 'url': 'https://video.twimg.com/ext_tw_video/7862048853
18561792/pu/vid/320x180/JuNJDqr1KHqoP83N.mp4'}, {'content_type': 'applicatio
n/x-mpegURL', 'url': 'https://video.twimg.com/ext_tw_video/78620488531856179
2/pu/pl/IugUNii3a7lmjApS.m3u8'}]}, 'additional_media_info': {'monetizable': F
alse}}]}, 'source': '<a href="http://twitter.com/download/iphone" rel="nofoll
ow">Twitter for iPhone</a>', 'in_reply_to_status_id': None, 'in_reply_to_stat
us_id_str': None, 'in_reply_to_user_id': None, 'in_reply_to_user_id_str': Non
e, 'in_reply_to_screen_name': None, 'user': {'id': 25073877, 'id_str': '25073
877', 'name': 'Donald J. Trump', 'screen_name': 'realDonaldTrump', 'locatio
n': 'Washington, DC', 'description': '45th President of the United States of
Americaus', 'url': None, 'entities': {'description': {'urls': []}}, 'protecte
d': False, 'followers_count': 35307313, 'friends_count': 45, 'listed_count':
74225, 'created_at': 'Wed Mar 18 13:46:38 +0000 2009', 'favourites_count': 1
2, 'utc_offset': -14400, 'time_zone': 'Eastern Time (US & Canada)', 'geo_enab
led': True, 'verified': True, 'statuses_count': 35480, 'lang': 'en', 'contrib
utors_enabled': False, 'is_translator': False, 'is_translation_enabled': Tru
e, 'profile_background_color': '6D5C18', 'profile_background_image_url': 'htt
p://pbs.twimg.com/profile_background_images/530021613/trump_scotland__43_of_7
0_cc.jpg', 'profile_background_image_url_https': 'https://pbs.twimg.com/profi
le_background_images/530021613/trump_scotland__43_of_70_cc.jpg', 'profile_bac
kground_tile': True, 'profile_image_url': 'http://pbs.twimg.com/profile_image
s/874276197357596672/kUuht00m_normal.jpg', 'profile_image_url_https': 'http
s://pbs.twimg.com/profile_images/874276197357596672/kUuht00m_normal.jpg', 'pr
ofile_banner_url': 'https://pbs.twimg.com/profile_banners/25073877/150191663
4', 'profile_link_color': '1B95E0', 'profile_sidebar_border_color': 'BDDCAD',
'profile_sidebar_fill_color': 'C5CEC0', 'profile_text_color': '333333', 'prof
ile_use_background_image': True, 'has_extended_profile': False, 'default_prof
ile': False, 'default_profile_image': False, 'following': False, 'follow_requ
est_sent': False, 'notifications': False, 'translator_type': 'regular'}, 'ge
o': None, 'coordinates': None, 'place': {'id': '4ec01c9dbc693497', 'url': 'ht
tps://api.twitter.com/1.1/geo/id/4ec01c9dbc693497.json', 'place_type': 'admi
n', 'name': 'Florida', 'full_name': 'Florida, USA', 'country_code': 'US', 'co
untry': 'United States', 'contained_within': [], 'bounding_box': {'type': 'Po
lygon', 'coordinates': [[[-87.634643, 24.396308], [-79.974307, 24.396308], [-
79.974307, 31.001056], [-87.634643, 31.001056]]]}, 'attributes': {}}, 'contri
butors': None, 'is_quote_status': False, 'retweet_count': 24915, 'favorite_co
unt': 42242, 'favorited': False, 'retweeted': False, 'possibly_sensitive': Fa
lse, 'lang': 'en'}
```

As a dictionary we can also list the keys:

```
In [92]: old_trump_tweets[0].keys()
```

```
Out[92]: dict_keys(['created_at', 'id', 'id_str', 'text', 'truncated', 'entities', 'ex
tended_entities', 'source', 'in_reply_to_status_id', 'in_reply_to_status_id_s
tr', 'in_reply_to_user_id', 'in_reply_to_user_id_str', 'in_reply_to_screen_na
me', 'user', 'geo', 'coordinates', 'place', 'contributors', 'is_quote_statu
s', 'retweet_count', 'favorite_count', 'favorited', 'retweeted', 'possibly_se
nsitive', 'lang'])
```

Since we're giving you a zipfile of old tweets, you may wonder why we didn't just give you a zipfile of ALL tweets and save you the trouble of creating a Twitter developer account. The reason is that we wanted you to see what it's like to collect data from the real world on your own. It can be a pain!

And for those of you that never got your developer accounts, you can see it can be even more of a pain that we expected. Sorry to anybody that wasted a bunch of time trying to get things working.

## Question 2

Merge the `old_trump_tweets` and the `trump_tweets` we downloaded from twitter into one giant list of tweets.

**Important:** There may be some overlap so be sure to eliminate duplicate tweets.
**Hint:** the `id` of a tweet is always unique.

```
In [93]: ### BEGIN SOLUTION
         all_tweets = old_trump_tweets + trump_tweets #TODO
         ### END SOLUTION
```

In [94]:

```
all_tweets[0]
```

```
Out[94]: {'created_at': 'Wed Oct 12 14:00:48 +0000 2016',
          'id': 786204978629185536,
          'id_str': '786204978629185536',
          'text': 'PAY TO PLAY POLITICS. \n#CrookedHillary https://t.co/wjsl8ITVvk',
          'truncated': False,
          'entities': {'hashtags': [{'text': 'CrookedHillary', 'indices': [23, 38]}],
           'symbols': [],
           'user_mentions': [],
           'urls': [],
           'media': [{'id': 786204885318561792,
             'id_str': '786204885318561792',
             'indices': [39, 62],
             'media_url': 'http://pbs.twimg.com/ext_tw_video_thumb/786204885318561792/
         pu/img/XqMoixLm83FzkAbn.jpg',
             'media_url_https': 'https://pbs.twimg.com/ext_tw_video_thumb/786204885318
         561792/pu/img/XqMoixLm83FzkAbn.jpg',
             'url': 'https://t.co/wjsl8ITVvk',
             'display_url': 'pic.twitter.com/wjsl8ITVvk',
             'expanded_url': 'https://twitter.com/realDonaldTrump/status/7862049786291
         85536/video/1',
             'type': 'photo',
             'sizes': {'thumb': {'w': 150, 'h': 150, 'resize': 'crop'},
              'medium': {'w': 600, 'h': 338, 'resize': 'fit'},
              'small': {'w': 340, 'h': 191, 'resize': 'fit'},
              'large': {'w': 1024, 'h': 576, 'resize': 'fit'}}}]},
          'extended_entities': {'media': [{'id': 786204885318561792,
             'id_str': '786204885318561792',
             'indices': [39, 62],
             'media_url': 'http://pbs.twimg.com/ext_tw_video_thumb/786204885318561792/
         pu/img/XqMoixLm83FzkAbn.jpg',
             'media_url_https': 'https://pbs.twimg.com/ext_tw_video_thumb/786204885318
         561792/pu/img/XqMoixLm83FzkAbn.jpg',
             'url': 'https://t.co/wjsl8ITVvk',
             'display_url': 'pic.twitter.com/wjsl8ITVvk',
             'expanded_url': 'https://twitter.com/realDonaldTrump/status/7862049786291
         85536/video/1',
             'type': 'video',
             'sizes': {'thumb': {'w': 150, 'h': 150, 'resize': 'crop'},
              'medium': {'w': 600, 'h': 338, 'resize': 'fit'},
              'small': {'w': 340, 'h': 191, 'resize': 'fit'},
              'large': {'w': 1024, 'h': 576, 'resize': 'fit'}},
             'video_info': {'aspect_ratio': [16, 9],
              'duration_millis': 30106,
              'variants': [{'bitrate': 832000,
                'content_type': 'video/mp4',
                'url': 'https://video.twimg.com/ext_tw_video/786204885318561792/pu/vi
         d/640x360/6vt24D3ZQSvYuDqe.mp4'},
               {'bitrate': 2176000,
                'content_type': 'video/mp4',
                'url': 'https://video.twimg.com/ext_tw_video/786204885318561792/pu/vi
         d/1280x720/rSbgQdvR9TPIlRWr.mp4'},
               {'bitrate': 320000,
                'content_type': 'video/mp4',
                'url': 'https://video.twimg.com/ext_tw_video/786204885318561792/pu/vi
         d/320x180/JuNJDqr1KHqoP83N.mp4'},
               {'content_type': 'application/x-mpegURL',
                'url': 'https://video.twimg.com/ext_tw_video/786204885318561792/pu/pl/
IugUNii3a7lmjApS.m3u8'}]},
    'additional_media_info': {'monetizable': False}}]},
 'source': '<a href="http://twitter.com/download/iphone" rel="nofollow">Twitt
er for iPhone</a>',
 'in_reply_to_status_id': None,
 'in_reply_to_status_id_str': None,
 'in_reply_to_user_id': None,
 'in_reply_to_user_id_str': None,
 'in_reply_to_screen_name': None,
 'user': {'id': 25073877,
  'id_str': '25073877',
  'name': 'Donald J. Trump',
  'screen_name': 'realDonaldTrump',
  'location': 'Washington, DC',
  'description': '45th President of the United States of Americaus',
  'url': None,
  'entities': {'description': {'urls': []}},
  'protected': False,
  'followers_count': 35307313,
  'friends_count': 45,
  'listed_count': 74225,
  'created_at': 'Wed Mar 18 13:46:38 +0000 2009',
  'favourites_count': 12,
  'utc_offset': -14400,
  'time_zone': 'Eastern Time (US & Canada)',
  'geo_enabled': True,
  'verified': True,
  'statuses_count': 35480,
  'lang': 'en',
  'contributors_enabled': False,
  'is_translator': False,
  'is_translation_enabled': True,
  'profile_background_color': '6D5C18',
  'profile_background_image_url': 'http://pbs.twimg.com/profile_background_im
ages/530021613/trump_scotland__43_of_70_cc.jpg',
  'profile_background_image_url_https': 'https://pbs.twimg.com/profile_backgr
ound_images/530021613/trump_scotland__43_of_70_cc.jpg',
  'profile_background_tile': True,
  'profile_image_url': 'http://pbs.twimg.com/profile_images/87427619735759667
2/kUuht00m_normal.jpg',
  'profile_image_url_https': 'https://pbs.twimg.com/profile_images/8742761973
57596672/kUuht00m_normal.jpg',
  'profile_banner_url': 'https://pbs.twimg.com/profile_banners/25073877/15019
16634',
  'profile_link_color': '1B95E0',
  'profile_sidebar_border_color': 'BDDCAD',
  'profile_sidebar_fill_color': 'C5CEC0',
  'profile_text_color': '333333',
  'profile_use_background_image': True,
  'has_extended_profile': False,
  'default_profile': False,
  'default_profile_image': False,
  'following': False,
  'follow_request_sent': False,
  'notifications': False,
  'translator_type': 'regular'},
 'geo': None,
 'coordinates': None,
 'place': {'id': '4ec01c9dbc693497',
  'url': 'https://api.twitter.com/1.1/geo/id/4ec01c9dbc693497.json',
  'place_type': 'admin',
  'name': 'Florida',
  'full_name': 'Florida, USA',
  'country_code': 'US',
  'country': 'United States',
  'contained_within': [],
  'bounding_box': {'type': 'Polygon',
   'coordinates': [[[-87.634643, 24.396308],
     [-79.974307, 24.396308],
     [-79.974307, 31.001056],
     [-87.634643, 31.001056]]]},
  'attributes': {}},
 'contributors': None,
 'is_quote_status': False,
 'retweet_count': 24915,
 'favorite_count': 42242,
 'favorited': False,
 'retweeted': False,
 'possibly_sensitive': False,
 'lang': 'en'}
```

In [95]:

```
assert len(all_tweets) > len(trump_tweets)
assert len(all_tweets) > len(old_trump_tweets)
### BEGIN HIDDEN TESTS
assert len(set([t['id'] for t in all_tweets])) <= len([t['id'] for t in all_tweets])
### END HIDDEN TESTS
```

## Question 3

Construct a DataFrame called `trump` containing all the tweets stored in `all_tweets`. The index of the dataframe should be the ID of each tweet (looks something like `907698529606541312`). It should have these columns:

- `time`: The time the tweet was created encoded as a datetime object. (Use `pd.to_datetime` to encode the timestamp.)
- `source`: The source device of the tweet.
- `text`: The text of the tweet.
- `retweet_count`: The retweet count of the tweet.

Finally, **the resulting dataframe should be sorted by the index.**

**Warning:** *Some tweets will store the text in the `text` field and other will use the `full_text` field.*

In [96]:

```
### BEGIN SOLUTION
trump = pd.DataFrame({'time': pd.to_datetime([tweet['created_at'] for tweet in all_tweets]),
                      'source': [tweet['source'] for tweet in all_tweets],
                      'text': [tweet['text'] if "text" in tweet else tweet['full_text'] for tweet in all_tweets],
                      'retweet_count': [tweet["retweet_count"] for tweet in all_tweets], },
                     index=[tweet['id'] for tweet in all_tweets],
                     columns=['time', 'source', 'text', 'retweet_count'],
                    ).sort_index()

trump.head(10)
#TODO
### END SOLUTION
```

Out[96]:

| | time | source | tex |
|---|---|---|---|
| **690171032150237184** | 2016-01-21 13:56:11+00:00 | <a href="http://twitter.com/download/android" rel="nofollow">Twitter for Android</a> | "@bigop1 @realDonaldTrump @SarahPalinUSA https://t.co/3kYQGqeVyD |
| **690171403388104704** | 2016-01-21 13:57:39+00:00 | <a href="http://twitter.com/download/android" rel="nofollow">Twitter for Android</a> | "@AmericanAsPie @glennbeck @SarahPalinUSA Remember when Glenn gave out gifts to ILLEGAL ALIENS at crossing the border? Me too! |
| **690173226341691392** | 2016-01-21 14:04:54+00:00 | <a href="http://twitter.com/download/android" rel="nofollow">Twitter for Android</a> | So sad that @CNN and many others refused to show the massive crowd at the arena yesterday in Oklahoma. Dishonest reporting |
| **690176882055114758** | 2016-01-21 14:19:26+00:00 | <a href="http://twitter.com/download/android" rel="nofollow">Twitter for Android</a> | Sad sack @JebBush has just done another ad on me, with special interest money, saying I won't beat Hillary - I WILL. But he can't beat me |
| **690180284189310976** | 2016-01-21 14:32:57+00:00 | <a href="http://twitter.com/download/android" rel="nofollow">Twitter for Android</a> | Low energy candidate @JebBush has wasted $80 million on his failed presidential campaign. Millions spent on me. He should go home and relax |
| **690271688127213568** | 2016-01-21 20:36:09+00:00 | <a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a> | New Day on CNN treats me very badly. @AlisynCamerota is a disaster. Not going to watch anymore |
| **690272687168458754** | 2016-01-21 20:40:07+00:00 | <a href="http://twitter.com/download/android" rel="nofollow">Twitter for Android</a> | Happy birthday to my friend, the great @jacknicklaus - a totally special guy |
| **690313350278819840** | 2016-01-21 23:21:42+00:00 | <a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a> | Thank you, Iowa #Trump2016 https://t.co/ryhEheTLqN |
| **690315202261155840** | 2016-01-21 23:29:04+00:00 | <a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a> | Thank you! #Trump2016 https://t.co/pcdmyIO1Z |
| **690315366564626433** | 2016-01-21 23:29:43+00:00 | <a href="http://twitter.com/download/iphone" rel="nofollow">Twitter for iPhone</a> | Thank you, New Hampshire!\n#Trump2016 https://t.co/TG9oZKly4 |

In [97]:
```
assert isinstance(trump, pd.DataFrame)
assert trump.shape[0] < 11000
assert trump.shape[1] >= 4
assert 831846101179314177 in trump.index
assert 753063644578144260 in trump.index
assert all(col in trump.columns for col in ['time', 'source', 'text', 'retweet
_count'])
# If you fail these tests, you probably tried to use __dict__ or _json to read
in the tweets
assert np.sometrue([('Twitter for iPhone' in s) for s in trump['source'].uniqu
e()])
assert isinstance(trump['time'].dtype, pd.core.dtypes.dtypes.DatetimeTZDtype)
assert trump['text'].dtype == np.dtype('O')
assert trump['retweet_count'].dtype == np.dtype('int64')
```

# Question 4: Tweet Source Analysis

In the following questions, we are going to find out the charateristics of Trump tweets and the devices used for the tweets.

First let's examine the source field:

In [98]:
```
trump['source'].unique()
```

Out[98]:
```
array(['<a href="http://twitter.com/download/android" rel="nofollow">Twitter
for Android</a>',
       '<a href="http://twitter.com/download/iphone" rel="nofollow">Twitter f
or iPhone</a>',
       '<a href="http://twitter.com" rel="nofollow">Twitter Web Client</a>',
       '<a href="https://mobile.twitter.com" rel="nofollow">Mobile Web (M5)</
a>',
       '<a href="http://instagram.com" rel="nofollow">Instagram</a>',
       '<a href="http://twitter.com/#!/download/ipad" rel="nofollow">Twitter
for iPad</a>',
       '<a href="https://studio.twitter.com" rel="nofollow">Media Studio</a
>',
       '<a href="https://periscope.tv" rel="nofollow">Periscope</a>',
       '<a href="https://ads.twitter.com" rel="nofollow">Twitter Ads</a>'],
      dtype=object)
```

# Question 4a

Remove the HTML tags from the source field.

**Hint:** Use `trump['source'].str.replace` and your favorite regular expression.

In [99]:

```
### BEGIN SOLUTION
new = trump['source'].str.replace(r'<[^>]*>', "")
trump['source'] = new
#TODO

### END SOLUTION
```

In [100]:

```
from datetime import datetime, timezone
ELEC_DATE = datetime(2016, 11, 8, tzinfo=timezone.utc)
INAUG_DATE = datetime(2017, 1, 20, tzinfo=timezone.utc)
assert set(trump[(trump['time'] > ELEC_DATE) & (trump['time'] < INAUG_DATE) ][
'source'].unique()) == set(['Twitter Ads',
 'Twitter Web Client',
 'Twitter for Android',
 'Twitter for iPhone'])
```

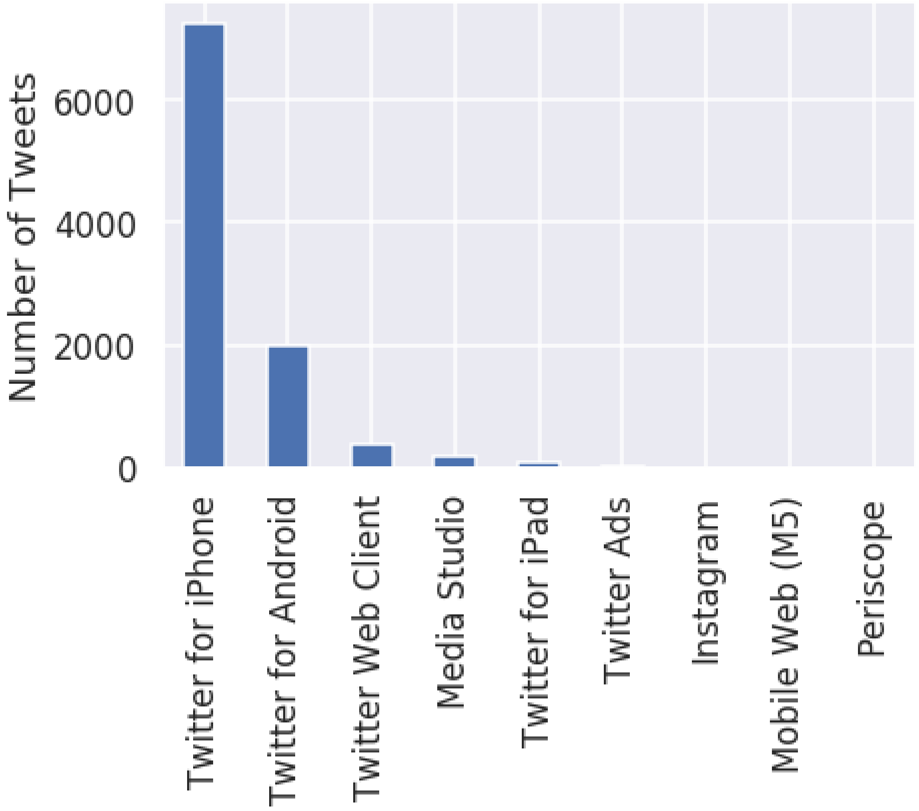
We can see in the following plot that there are two device types that are more commonly used

In [101]:

```
trump['source'].value_counts().plot(kind="bar")
plt.ylabel("Number of Tweets")
```

Out[101]:

```
Text(0, 0.5, 'Number of Tweets')
```



## Question 4b

Is there a difference between his Tweet behavior across these devices? We will attempt to answer this question in our subsequent analysis.

First, we'll take a look at whether Trump's tweets from an Android come at different times than his tweets from an iPhone. Note that Twitter gives us his tweets in the UTC timezone (https://www.wikiwand.com/en/List_of_UTC_time_offsets) (notice the `+0000` in the first few tweets)

In [102]:
```
for t in trump_tweets[0:3]:
    print(t['created_at'])
```

```
Tue Oct 16 16:22:11 +0000 2018
Tue Oct 16 16:18:08 +0000 2018
Tue Oct 16 15:26:33 +0000 2018
```

We'll convert the tweet times to US Eastern Time, the timezone of New York and Washington D.C., since those are the places we would expect the most tweet activity from Trump.

In [103]:

```
trump['est_time'] = (
    trump['time'].dt.tz_convert("EST") # Convert to Eastern Time
    # If your data frame is, for some reason, not timezone-aware
    # you might need the below two lines instead:
    #  trump['time'].dt.tz_localize("UTC") # Set initial timezone to UTC
    # .trump['time'].dt.tz_convert("EST") # Convert to Eastern Time
)
trump.head()
```

Out[103]:

| | time | source | text | retweet_count | est_time |
|---|---|---|---|---|---|
| **690171032150237184** | 2016-01-21 13:56:11+00:00 | Twitter for Android | "@bigop1: @realDonaldTrump @SarahPalinUSA https://t.co/3kYQGqeVyD" | 1059 | 2016-01-21 08:56:11-05:00 |
| **690171403388104704** | 2016-01-21 13:57:39+00:00 | Twitter for Android | "@AmericanAsPie: @glennbeck @SarahPalinUSA Remember when Glenn gave out gifts to ILLEGAL ALIENS at crossing the border? Me too!" | 1339 | 2016-01-21 08:57:39-05:00 |
| **690173226341691392** | 2016-01-21 14:04:54+00:00 | Twitter for Android | So sad that @CNN and many others refused to show the massive crowd at the arena yesterday in Oklahoma. Dishonest reporting! | 2006 | 2016-01-21 09:04:54-05:00 |
| **690176882055114758** | 2016-01-21 14:19:26+00:00 | Twitter for Android | Sad sack @JebBush has just done another ad on me, with special interest money, saying I won't beat Hillary - I WILL. But he can't beat me. | 2266 | 2016-01-21 09:19:26-05:00 |
| **690180284189310976** | 2016-01-21 14:32:57+00:00 | Twitter for Android | Low energy candidate @JebBush has wasted $80 million on his failed presidential campaign. Millions spent on me. He should go home and relax! | 2886 | 2016-01-21 09:32:57-05:00 |

**What you need to do:**

Add a column called `hour` to the `trump` table which contains the hour of the day as floating point number computed by:

$$\text{hour} + \frac{\text{minute}}{60} + \frac{\text{second}}{60^2}$$

In [104]:

```
trump['hour'] = (
    trump['est_time'].dt.hour + trump['est_time'].dt.minute/60 +
    trump['est_time'].dt.second/(60*60)
)
trump.head()
```

Out[104]:

| | time | source | text | retweet_count | est_time | |
|---|---|---|---|---|---|---|
| **690171032150237184** | 2016-01-21 13:56:11+00:00 | Twitter for Android | "@bigop1: @realDonaldTrump @SarahPalinUSA https://t.co/3kYQGqeVyD" | 1059 | 2016-01-21 08:56:11-05:00 | 8 |
| **690171403388104704** | 2016-01-21 13:57:39+00:00 | Twitter for Android | "@AmericanAsPie: @glennbeck @SarahPalinUSA Remember when Glenn gave out gifts to ILLEGAL ALIENS at crossing the border? Me too!" | 1339 | 2016-01-21 08:57:39-05:00 | 8 |
| **690173226341691392** | 2016-01-21 14:04:54+00:00 | Twitter for Android | So sad that @CNN and many others refused to show the massive crowd at the arena yesterday in Oklahoma. Dishonest reporting! | 2006 | 2016-01-21 09:04:54-05:00 | 9 |
| **690176882055114758** | 2016-01-21 14:19:26+00:00 | Twitter for Android | Sad sack @JebBush has just done another ad on me, with special interest money, saying I won't beat Hillary - I WILL. But he can't beat me. | 2266 | 2016-01-21 09:19:26-05:00 | 9 |
| **690180284189310976** | 2016-01-21 14:32:57+00:00 | Twitter for Android | Low energy candidate @JebBush has wasted $80 million on his failed presidential campaign. Millions spent on me. He should go home and relax! | 2886 | 2016-01-21 09:32:57-05:00 | 9 |

In [105]:

```
assert np.isclose(trump.loc[690171032150237184]['hour'], 8.93639)
```

## Question 4c

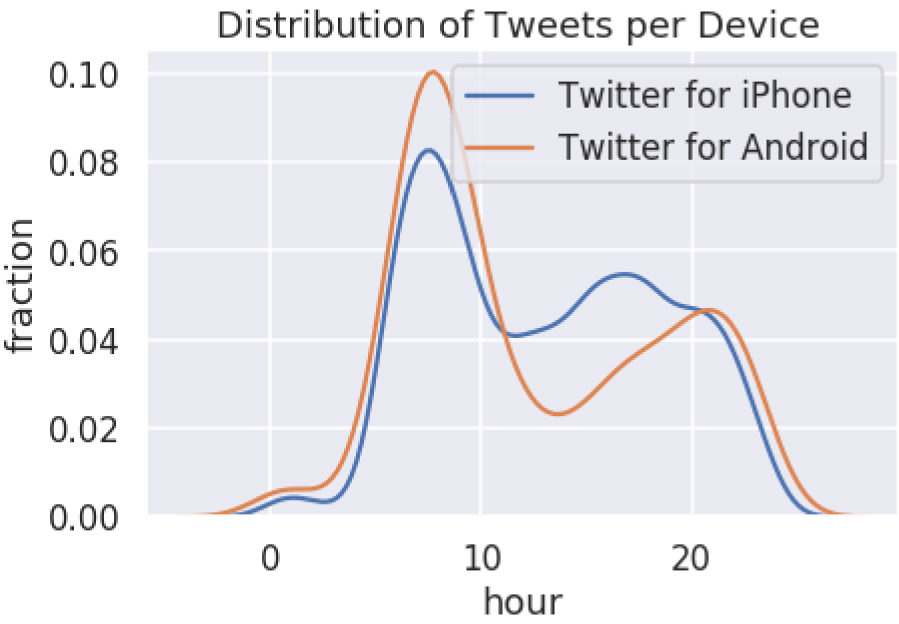
Use this data along with the seaborn `distplot` function to examine the distribution over hours of the day in eastern time that trump tweets on each device for the 2 most commonly used devices. Your plot should look similar to the following.

In [106]:

```
### make your plot here

device = trump['source'].value_counts()
device = device[0:2]
device
devices = device.index
devices
for x in devices:
    sns.distplot(trump[trump.source == x]['hour'], label=x, hist = False)
plt.ylabel('fraction')
plt.title('Distribution of Tweets per Device ')
```

Out[106]:

```
Text(0.5, 1.0, 'Distribution of Tweets per Device ')
```



## Question 4d

According to [this Verge article (https://www.theverge.com/2017/3/29/15103504/donald-trump-iphone-using-switched-android)](https://www.theverge.com/2017/3/29/15103504/donald-trump-iphone-using-switched-android), Donald Trump switched from an Android to an iPhone sometime in March 2017.

Create a figure identical to your figure from 4c, except that you should show the results only from 2016. If you get stuck consider looking at the `year_fraction` function from the next problem.

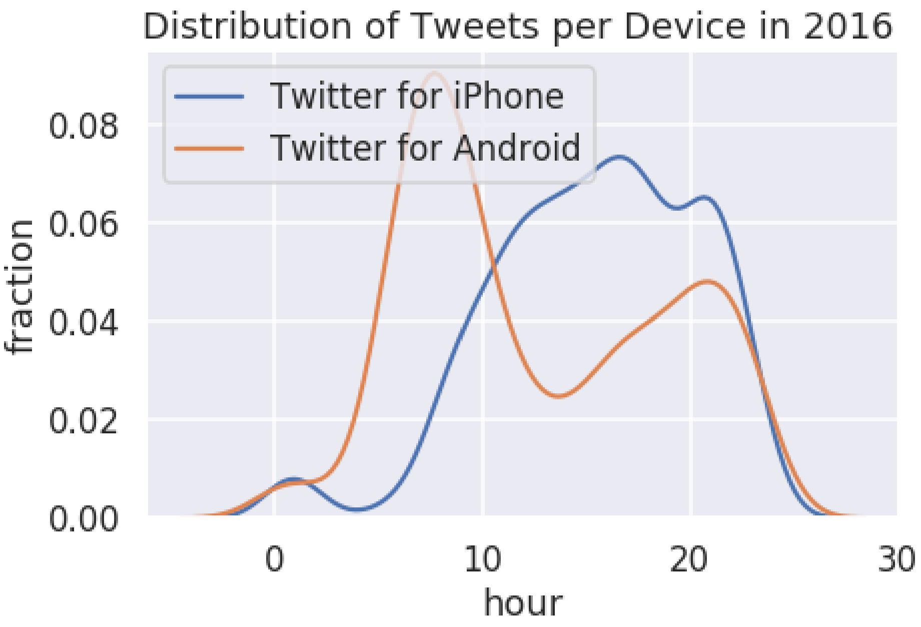
During the campaign, it was theorized that Donald Trump's tweets from Android were written by him personally, and the tweets from iPhone were from his staff. Does your figure give support to this theory?

In [107]:

```
import datetime
def year_fraction(date):
    start = datetime.date(date.year, 1, 1).toordinal()
    year_length = datetime.date(date.year+1, 1, 1).toordinal() - start
    return date.year + float(date.toordinal() - start) / year_length


trump['year'] = trump['time'].apply(year_fraction)

device = trump['source'].value_counts().head(2)
devices = device.index
devices
for x in devices:
    sns.distplot(trump[(trump.source == x) & (trump['year']%2016 < 1)]['hou
r'], label=x, hist = False)
plt.ylabel('fraction')
plt.title('Distribution of Tweets per Device in 2016 ')### make your plot here
### BEGIN SOLUTION
#TODO
### END SOLUTION
```

Out[107]:

```
Text(0.5, 1.0, 'Distribution of Tweets per Device in 2016 ')
```



Yes, our figure shows that the Android tweets were typically very late at night when Donald Trump is known to tweet, and when paid staff are unlikely to be posting.

## Question 5

Let's now look at which device he has used over the entire time period of this dataset.

To examine the distribution of dates we will convert the date to a fractional year that can be plotted as a distribution.

(Code borrowed from https://stackoverflow.com/questions/6451655/python-how-to-convert-datetime-dates-to-decimal-years (https://stackoverflow.com/questions/6451655/python-how-to-convert-datetime-dates-to-decimal-years))

In [108]:

```
import datetime
def year_fraction(date):
    start = datetime.date(date.year, 1, 1).toordinal()
    year_length = datetime.date(date.year+1, 1, 1).toordinal() - start
    return date.year + float(date.toordinal() - start) / year_length


trump['year'] = trump['time'].apply(year_fraction)
```

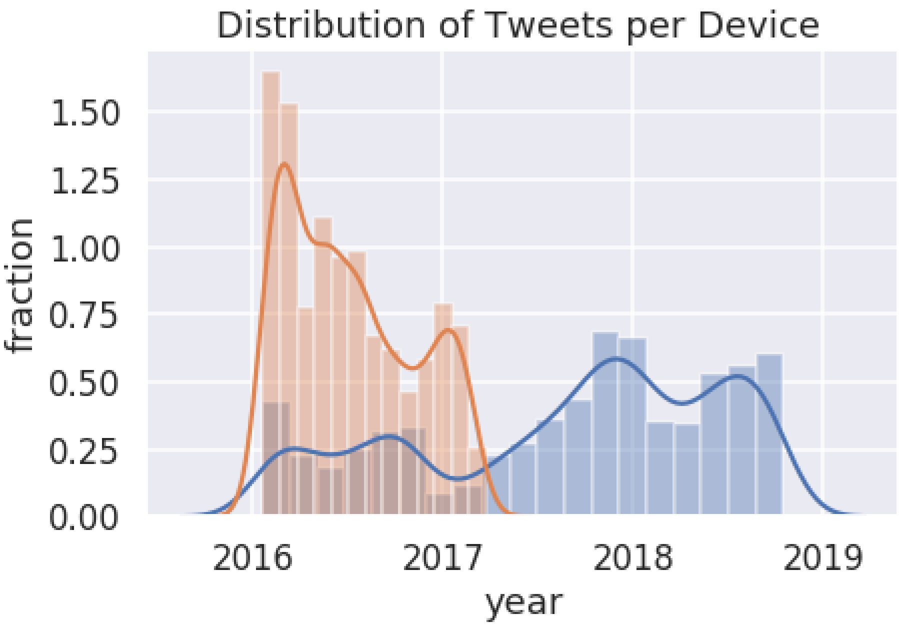
Use the `sns.distplot` to overlay the distributions of the 2 most frequently used web technologies over the years. Your final plot should look like:

In [145]:

```
### BEGIN SOLUTION
device = trump['source'].value_counts()
device = device[0:2]
devices = device.index
devices
for temp in devices:
    sns.distplot(trump[trump.source == temp]['year'], label=temp)
plt.ylabel('fraction')
plt.title('Distribution of Tweets per Device ')
#TODO
### END SOLUTION
```

Out[145]:

```
Text(0.5, 1.0, 'Distribution of Tweets per Device ')
```



## Question 6: Sentiment Analysis

It turns out that we can use the words in Trump's tweets to calculate a measure of the sentiment of the tweet. For example, the sentence "I love America!" has positive sentiment, whereas the sentence "I hate taxes!" has a negative sentiment. In addition, some words have stronger positive / negative sentiment than others: "I love America." is more positive than "I like America."

We will use the VADER (Valence Aware Dictionary and sEntiment Reasoner) (https://github.com/cjhutto/vaderSentiment) lexicon to analyze the sentiment of Trump's tweets. VADER is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media which is great for our usage.

The VADER lexicon gives the sentiment of individual words. Run the following cell to show the first few rows of the lexicon:

```
In [110]: print(''.join(open("vader_lexicon.txt").readlines()[-100:]))
```

```
withdrawal      0.1     1.57797 [1, -1, 0, -2, -2, 2, -1, 1, 0, 3]
woe     -1.8    0.6     [-3, -2, -2, -2, -1, -1, -2, -1, -2, -2]
woebegone       -2.6    0.66332 [-3, -2, -3, -2, -2, -4, -3, -2, -2, -3]
woebegoneness   -1.1    1.37477 [-3, 0, -1, 1, -1, -4, 0, -1, -1, -1]
woeful  -1.9    0.83066 [-1, -2, -2, -1, -3, -3, -1, -2, -1, -3]
woefully        -1.7    1.48661 [-1, -3, -2, 1, -3, -3, -2, -2, 1, -3]
woefulness      -2.1    0.7     [-3, -2, -2, -1, -2, -3, -3, -1, -2, -2]
woes    -1.9    0.83066 [-2, -2, -2, -1, -2, -3, -3, 0, -2, -2]
woesome -1.2    1.6     [-2, -3, -2, -1, 0, 3, -2, -2, -1, -2]
won     2.7     0.9     [3, 4, 2, 2, 2, 4, 4, 2, 2, 2]
wonderful       2.7     0.78102 [2, 3, 3, 2, 4, 2, 2, 3, 4, 2]
wonderfully     2.9     0.83066 [1, 3, 3, 4, 3, 2, 3, 3, 4, 3]
wonderfulness   2.9     0.53852 [3, 2, 3, 3, 3, 3, 3, 2, 4, 3]
woo     2.1     1.37477 [4, 2, 1, 3, 2, 2, -1, 2, 2, 4]
woohoo  2.3     1.1     [3, 3, 1, 4, 4, 2, 1, 1, 2, 2]
woot    1.8     1.07703 [2, 0, 2, 2, 2, 2, 0, 4, 2, 2]
worn    -1.2    0.4     [-1, -1, -1, -1, -1, -1, -2, -1, -2, -1]
worried -1.2    0.74833 [-1, -1, -1, -1, -1, -2, -3, 0, -1, -1]
worriedly       -2.0    0.44721 [-2, -2, -3, -2, -2, -2, -2, -1, -2, -2]
worrier -1.8    0.6     [-2, -2, -1, -2, -1, -3, -2, -2, -1, -2]
worriers        -1.7    0.45826 [-2, -1, -2, -2, -2, -2, -1, -2, -1, -2]
worries -1.8    0.6     [-2, -2, -1, -2, -1, -2, -2, -3, -1, -2]
worriment       -1.5    0.67082 [-1, -2, -1, -1, -1, -2, -1, -3, -1, -2]
worriments      -1.9    0.7     [-2, -1, -2, -3, -1, -2, -3, -1, -2, -2]
worrisome       -1.7    0.64031 [-1, -1, -1, -2, -1, -2, -3, -2, -2, -2]
worrisomely     -2.0    0.63246 [-1, -2, -1, -2, -2, -3, -2, -2, -3, -2]
worrisomeness   -1.9    0.53852 [-2, -2, -3, -1, -2, -2, -2, -1, -2, -2]
worrit  -2.1    0.53852 [-2, -2, -1, -2, -2, -3, -3, -2, -2, -2]
worrits -1.2    0.9798  [-1, -2, -2, -1, 0, 0, -1, -3, 0, -2]
worry   -1.9    0.7     [-2, -3, -1, -3, -1, -2, -1, -2, -2, -2]
worrying        -1.4    0.66332 [-2, -1, -2, -2, -1, 0, -1, -1, -2, -2]
worrywart       -1.8    0.9798  [-2, -2, -2, -1, -1, -1, -1, -3, -1, -4]
worrywarts      -1.5    0.5     [-2, -1, -2, -2, -2, -1, -1, -1, -2, -1]
worse   -2.1    0.83066 [-2, -2, -1, -3, -4, -2, -1, -2, -2, -2]
worsen  -2.3    0.78102 [-4, -3, -1, -2, -2, -2, -2, -3, -2, -2]
worsened        -1.9    1.22066 [-2, -2, -2, -1, -2, -2, -4, 1, -3, -2]
worsening       -2.0    0.44721 [-2, -3, -2, -2, -2, -2, -1, -2, -2, -2]
worsens -2.1    0.53852 [-2, -2, -2, -2, -1, -2, -2, -3, -3, -2]
worser  -2.0    0.89443 [-2, -2, -4, -1, -2, -2, -2, -3, -1, -1]
worship 1.2     1.07703 [1, 0, 0, 1, 3, 0, 2, 3, 1, 1]
worshiped       2.4     1.0198  [1, 2, 4, 3, 4, 1, 2, 3, 2, 2]
worshiper       1.0     1.0     [0, 0, 2, 3, 0, 2, 1, 1, 1, 0]
worshipers      0.9     0.83066 [0, 0, 0, 2, 1, 1, 1, 2, 2, 0]
worshipful      0.7     1.00499 [1, -1, 3, 1, 1, 1, 0, 0, 0, 1]
worshipfully    1.1     1.3     [0, 0, 0, 1, 3, 0, 3, 3, 1, 0]
worshipfulness  1.6     0.8     [3, 1, 2, 2, 1, 1, 3, 1, 1, 1]
worshiping      1.0     1.18322 [0, 3, 0, 3, 0, 1, 1, 2, 0, 0]
worshipless     -0.6    1.0198  [0, -1, -3, -1, -1, -1, 0, 0, 0, 1]
worshipped      2.7     0.78102 [3, 2, 3, 3, 1, 4, 2, 3, 3, 3]
worshipper      0.6     0.66332 [1, 1, 0, 0, 1, 0, 0, 2, 1, 0]
worshippers     0.8     0.87178 [0, 1, 0, 0, 3, 1, 1, 1, 0, 1]
worshipping     1.6     1.28062 [1, 3, 3, 3, 0, 3, 1, 0, 2, 0]
worships        1.4     1.11355 [2, 0, 1, 3, 2, 1, 0, 3, 2, 0]
worst   -3.1    1.04403 [-4, -4, -3, -1, -3, -4, -2, -2, -4, -4]
worth   0.9     0.9434  [0, 0, 1, 1, 2, 1, 1, 3, 0, 0]
worthless       -1.9    1.13578 [-3, -1, -3, -4, -1, -3, -1, -1, -1, -1]
worthwhile      1.4     0.4899  [1, 1, 1, 2, 1, 1, 2, 1, 2, 2]
```

```
worthy  1.9     0.53852 [2, 2, 2, 1, 1, 2, 2, 2, 3, 2]
wow     2.8     0.9798  [2, 3, 2, 4, 4, 3, 3, 2, 1, 4]
wowed   2.6     0.8     [3, 3, 4, 3, 2, 1, 3, 3, 2, 2]
wowing  2.5     0.67082 [2, 2, 3, 3, 2, 3, 4, 2, 2, 2]
wows    2.0     1.61245 [2, 3, 3, 3, 2, 1, -2, 1, 4, 3]
wowser  -1.1    2.02237 [-3, 3, 0, 2, -2, -1, -3, -2, -2, -3]
wowsers 1.0     2.14476 [0, -2, 4, 2, 3, 0, 1, 2, -3, 3]
wrathful        -2.7    0.64031 [-3, -2, -2, -3, -3, -2, -4, -2, -3, -3]
wreck   -1.9    0.7     [-1, -2, -3, -3, -2, -2, -2, -1, -1, -2]
wrong   -2.1    1.04403 [-2, -2, -2, -2, -4, -4, -1, -1, -1, -2]
wronged -1.9    0.53852 [-2, -2, -2, -2, -2, -1, -3, -2, -2, -1]
x-d     2.6     0.91652 [2, 3, 3, 4, 1, 2, 3, 4, 2, 2]
x-p     1.7     0.45826 [2, 2, 1, 2, 2, 1, 1, 2, 2, 2]
xd      2.8     0.87178 [3, 3, 4, 2, 3, 3, 1, 2, 4, 3]
xp      1.6     0.4899  [2, 2, 2, 1, 1, 1, 2, 2, 1, 2]
yay     2.4     1.0198  [1, 3, 3, 2, 2, 1, 4, 4, 2, 2]
yeah    1.2     0.6     [1, 1, 1, 2, 1, 1, 0, 2, 1, 2]
yearning        0.5     1.0247  [0, 1, 0, 1, 0, 3, 0, 1, -1, 0]
yeees   1.7     1.00499 [1, 3, 1, 2, 1, 1, 4, 2, 1, 1]
yep     1.2     0.4     [1, 1, 1, 1, 1, 1, 2, 2, 1, 1]
yes     1.7     0.78102 [1, 2, 2, 1, 1, 1, 3, 3, 1, 2]
youthful        1.3     0.45826 [1, 2, 1, 2, 1, 1, 1, 1, 2, 1]
yucky   -1.8    0.6     [-2, -1, -1, -2, -2, -1, -2, -2, -3, -2]
yummy   2.4     1.0198  [1, 2, 4, 3, 2, 2, 3, 1, 4, 2]
zealot  -1.9    1.04403 [-2, -3, -1, -2, -1, -3, -4, -1, -1, -1]
zealots -0.8    1.83303 [-1, -2, -1, -2, -2, 1, -2, 4, -1, -2]
zealous 0.5     1.43178 [2, -1, 2, 1, 0, 0, 3, 0, -2, 0]
{:      1.8     0.9798  [1, 3, 2, 2, 1, 1, 4, 2, 1, 1]
|-0     -1.2    0.74833 [0, -2, -1, -1, -1, -1, -1, -1, -1, -3]
|-:     -0.8    0.74833 [-1, -2, 0, -1, 0, -2, -1, -1, 0, 0]
|-:>    -1.6    0.4899  [-1, -2, -2, -2, -2, -1, -1, -2, -2, -1]
|-o     -1.2    0.9798  [-1, 0, -1, -1, -1, -1, -1, -4, -1, -1]
|:      -0.5    1.68819 [2, -3, -1, 0, -1, -1, -1, -2, -1, 3]
|;-)    2.2     1.32665 [4, 1, 1, 1, 3, 2, 4, 1, 4, 1]
|=      -0.4    1.56205 [2, -2, -1, 0, -1, -1, -1, -2, -1, 3]
|^:     -1.1    0.7     [-2, 0, -1, -1, 0, -1, -1, -2, -2, -1]
|o:     -0.9    0.53852 [-1, 0, -1, -2, -1, 0, -1, -1, -1, -1]
||-:    -2.3    0.45826 [-2, -2, -2, -3, -3, -3, -2, -2, -2, -2]
}:      -2.1    0.83066 [-1, -1, -3, -2, -3, -2, -2, -1, -3, -3]
}:(     -2.0    0.63246 [-3, -1, -2, -1, -3, -2, -2, -2, -2, -2]
}:)     0.4     1.42829 [1, 1, -2, 1, 2, -2, 1, -1, 2, 1]
}:-(    -2.1    0.7     [-2, -1, -2, -2, -2, -4, -2, -2, -2, -2]
}:-)    0.3     1.61555 [1, 1, -2, 1, -1, -3, 2, 2, 1, 1]
```

# Question 6a

As you can see, the lexicon contains emojis too! The first column of the lexicon is the *token*, or the word itself. The second column is the *polarity* of the word, or how positive / negative it is.

(How did they decide the polarities of these words? What are the other two columns in the lexicon? See the link above.)

Read in the lexicon into a DataFrame called `sent` . The index of the DF should be the tokens in the lexicon. `sent` should have one column: `polarity` : The polarity of each token.

In [164]:

```
### BEGIN SOLUTION

sent = pd.read_csv('vader_lexicon.txt', sep='\t', names=['tokens', 'polarity', 'temp', 'temp2'])
sent = sent.drop(columns=['temp', 'temp2'])
sent = sent.set_index('tokens')
sent.head()
```

Out[164]:

| | polarity |
|---|---|
| **tokens** | |
| **$:** | -1.5 |
| **%)** | -0.4 |
| **%-)** | -1.5 |
| **&-:** | -0.4 |
| **&:** | -0.7 |

In [165]:

```
assert isinstance(sent, pd.DataFrame)
assert sent.shape == (7517, 1)
assert list(sent.index[5000:5005]) == ['paranoids', 'pardon', 'pardoned', 'pardoning', 'pardons']
assert np.allclose(sent['polarity'].head(), [-1.5, -0.4, -1.5, -0.4, -0.7])
```

# Question 6b

Now, let's use this lexicon to calculate the overall sentiment for each of Trump's tweets. Here's the basic idea:

1. For each tweet, find the sentiment of each word.
2. Calculate the sentiment of each tweet by taking the sum of the sentiments of its words.

First, let's lowercase the text in the tweets since the lexicon is also lowercase. Set the `text` column of the `trump` DF to be the lowercased text of each tweet.

In [166]:
```
### BEGIN SOLUTION
trump.head()

trump['text'] = trump['text'].str.lower()
trump.head()
```

Out[166]:

| | time | source | text | retweet_count | est_time | |
|---|---|---|---|---|---|---|
| **690171032150237184** | 2016-01-21 13:56:11+00:00 | Twitter for Android | "@bigop1: @realdonaldtrump @sarahpalinusa https://t.co/3kyqgqevyd" | 1059 | 2016-01-21 08:56:11-05:00 | 8.93 |
| **690171403388104704** | 2016-01-21 13:57:39+00:00 | Twitter for Android | "@americanaspie: @glennbeck @sarahpalinusa remember when glenn gave out gifts to illegal aliens at crossing the border? me too!" | 1339 | 2016-01-21 08:57:39-05:00 | 8.96 |
| **690173226341691392** | 2016-01-21 14:04:54+00:00 | Twitter for Android | so sad that @cnn and many others refused to show the massive crowd at the arena yesterday in oklahoma. dishonest reporting! | 2006 | 2016-01-21 09:04:54-05:00 | 9.08 |
| **690176882055114758** | 2016-01-21 14:19:26+00:00 | Twitter for Android | sad sack @jebbush has just done another ad on me, with special interest money, saying i won't beat hillary - i will. but he can't beat me. | 2266 | 2016-01-21 09:19:26-05:00 | 9.32 |
| **690180284189310976** | 2016-01-21 14:32:57+00:00 | Twitter for Android | low energy candidate @jebbush has wasted $80 million on his failed presidential campaign. millions spent on me. he should go home and relax! | 2886 | 2016-01-21 09:32:57-05:00 | 9.54 |

In [167]:
```
assert trump['text'].loc[884740553040175104] == 'working hard to get the olymp
ics for the united states (l.a.). stay tuned!'
```

## Question 6c

Now, let's get rid of punctuation since it'll cause us to fail to match words. Create a new column called `no_punc` in the `trump` DF to be the lowercased text of each tweet with all punctuation replaced by a single space. We consider punctuation characters to be any character that isn't a Unicode word character or a whitespace character. You may want to consult the Python documentation on regexes for this problem.

(Why don't we simply remove punctuation instead of replacing with a space? See if you can figure this out by looking at the tweet data.)

In [168]:

```
# Save your regex in punct_re

### BEGIN SOLUTION
#TODO
import re
punct_re = r'[^\s\w]'
trump['no_punc'] = trump['text'].str.replace(punct_re, " ")
trump.head()
### END SOLUTION
```

Out[168]:

| | time | source | text | retweet_count | est_time | |
|---|---|---|---|---|---|---|
| **690171032150237184** | 2016-01-21 13:56:11+00:00 | Twitter for Android | "@bigop1: @realdonaldtrump @sarahpalinusa https://t.co/3kyqgqevyd" | 1059 | 2016-01-21 08:56:11-05:00 | 8.9 |
| **690171403388104704** | 2016-01-21 13:57:39+00:00 | Twitter for Android | "@americanaspie: @glennbeck @sarahpalinusa remember when glenn gave out gifts to illegal aliens at crossing the border? me too!" | 1339 | 2016-01-21 08:57:39-05:00 | 8.9 |
| **690173226341691392** | 2016-01-21 14:04:54+00:00 | Twitter for Android | so sad that @cnn and many others refused to show the massive crowd at the arena yesterday in oklahoma. dishonest reporting! | 2006 | 2016-01-21 09:04:54-05:00 | 9.0 |
| **690176882055114758** | 2016-01-21 14:19:26+00:00 | Twitter for Android | sad sack @jebbush has just done another ad on me, with special interest money, saying i won't beat hillary - i will. but he can't beat me. | 2266 | 2016-01-21 09:19:26-05:00 | 9.3 |
| **690180284189310976** | 2016-01-21 14:32:57+00:00 | Twitter for Android | low energy candidate @jebbush has wasted $80 million on his failed presidential campaign. millions spent on me. he should go home and relax! | 2886 | 2016-01-21 09:32:57-05:00 | 9.5 |

```
In [169]: assert isinstance(punct_re, str)
assert re.search(punct_re, 'this') is None
assert re.search(punct_re, 'this is ok') is None
assert re.search(punct_re, 'this is\nok') is None
assert re.search(punct_re, 'this is not ok.') is not None
assert re.search(punct_re, 'this#is#ok') is not None
assert re.search(punct_re, 'this^is ok') is not None
assert trump['no_punc'].loc[800329364986626048] == 'i watched parts of  nbcsnl saturday night live last night  it is a totally one sided  biased show   nothing funny at all  equal time for us '
assert trump['no_punc'].loc[894620077634592769] == 'on  purpleheartday i thank all the brave men and women who have sacrificed in battle for this great nation   usa   https   t co qmfdlslp6p'
# If you fail these tests, you accidentally changed the text column
assert trump['text'].loc[884740553040175104] == 'working hard to get the olympics for the united states (l.a.). stay tuned!'
```

## Question 6d:

Now, let's convert the tweets into what's called a *tidy format* (https://cran.r-project.org/web/packages/tidyr/vignettes/tidy-data.html) to make the sentiments easier to calculate. Use the `no_punc` column of `trump` to create a table called `tidy_format` . The index of the table should be the IDs of the tweets, repeated once for every word in the tweet. It has two columns:

1. `num` : The location of the word in the tweet. For example, if the tweet was "i love america", then the location of the word "i" is 0, "love" is 1, and "america" is 2.
2. `word` : The individual words of each tweet.

The first few rows of our `tidy_format` table look like:

| | num | word |
|---|---|---|
| **894661651760377856** | 0 | i |
| **894661651760377856** | 1 | think |
| **894661651760377856** | 2 | senator |
| **894661651760377856** | 3 | blumenthal |
| **894661651760377856** | 4 | should |

**Note that you'll get different results depending on when you pulled in the tweets.** However, you can double check that your tweet with ID `894661651760377856` has the same rows as ours. Our tests don't check whether your table looks exactly like ours.

As usual, try to avoid using any for loops. Our solution uses a chain of 5 methods on the 'trump' DF, albeit using some rather advanced Pandas hacking.

- **Hint 1:** Try looking at the `expand` argument to pandas' `str.split` .
- **Hint 2:** Try looking at the `stack()` method.
- **Hint 3:** Try looking at the `level` parameter of the `reset_index` method.

In [170]:

```
### BEGIN SOLUTION
tidy_format = pd.DataFrame(trump['no_punc'].str.split(expand=True).stack().reset_index(level=1).rename(columns={'level_1': 'num', 0: 'word'}))
#temp = trump['text'].str.split(expand=True).stack().reset_index(level=1)
#tidy_format = pd.DataFrame(temp)

#tidy_format['num'] = tidy_format['level_1']
#tidy_format['word'] = tidy_format[0]

#tidy_format = tidy_format.drop(columns=['level_1', 0])
#tidy_format.head()



### END SOLUTION
```

In [171]:

```
assert tidy_format.loc[894661651760377856].shape == (27, 2)
assert ' '.join(list(tidy_format.loc[894661651760377856]['word'])) == 'i think senator blumenthal should take a nice long vacation in vietnam where he lied about his service so he can at least say he was there'
```

## Question 6e:

Now that we have this table in the tidy format, it becomes much easier to find the sentiment of each tweet: we can join the table with the lexicon table.

Add a `polarity` column to the `trump` table. The `polarity` column should contain the sum of the sentiment polarity of each word in the text of the tweet.

**Hint** you will need to merge the `tidy_format` and `sent` tables and group the final answer.

In [172]:

```
### BEGIN SOLUTION
temp = (tidy_format.merge(sent, how='left', left_on='word', right_index=True))
temp = temp.reset_index()
temp.head()
temp = temp.loc[:, ['index', 'polarity']].groupby('index').sum().fillna(0)
temp.head()
trump['polarity'] = temp
trump.head()
#TODO
### END SOLUTION
```

Out[172]:

| | time | source | text | retweet_count | est_time | |
|---|---|---|---|---|---|---|
| **690171032150237184** | 2016-01-21 13:56:11+00:00 | Twitter for Android | "@bigop1: @realdonaldtrump @sarahpalinusa https://t.co/3kyqgqevyd" | 1059 | 2016-01-21 08:56:11-05:00 | 8.93 |
| **690171403388104704** | 2016-01-21 13:57:39+00:00 | Twitter for Android | "@americanaspie: @glennbeck @sarahpalinusa remember when glenn gave out gifts to illegal aliens at crossing the border? me too!" | 1339 | 2016-01-21 08:57:39-05:00 | 8.96 |
| **690173226341691392** | 2016-01-21 14:04:54+00:00 | Twitter for Android | so sad that @cnn and many others refused to show the massive crowd at the arena yesterday in oklahoma. dishonest reporting! | 2006 | 2016-01-21 09:04:54-05:00 | 9.08 |
| **690176882055114758** | 2016-01-21 14:19:26+00:00 | Twitter for Android | sad sack @jebbush has just done another ad on me, with special interest money, saying i won't beat hillary - i will. but he can't beat me. | 2266 | 2016-01-21 09:19:26-05:00 | 9.32 |
| **690180284189310976** | 2016-01-21 14:32:57+00:00 | Twitter for Android | low energy candidate @jebbush has wasted $80 million on his failed presidential campaign. millions spent on me. he should go home and relax! | 2886 | 2016-01-21 09:32:57-05:00 | 9.54 |

In [173]:
```
assert np.allclose(trump.loc[744701872456536064, 'polarity'], 8.4)
assert np.allclose(trump.loc[745304731346702336, 'polarity'], 2.5)
assert np.allclose(trump.loc[744519497764184064, 'polarity'], 1.7)
assert np.allclose(trump.loc[894661651760377856, 'polarity'], 0.2)
assert np.allclose(trump.loc[894620077634592769, 'polarity'], 5.4)
# If you fail this test, you dropped tweets with 0 polarity
assert np.allclose(trump.loc[744355251365511169, 'polarity'], 0.0)
```

Now we have a measure of the sentiment of each of his tweets! Note that this calculation is rather basic; you can read over the VADER readme to understand a more robust sentiment analysis.

Now, run the cells below to see the most positive and most negative tweets from Trump in your dataset:

In [174]:
```
print('Most negative tweets:')
for t in trump.sort_values('polarity').head()['text']:
    print('\n  ', t)
```

```
Most negative tweets:

   horrible and cowardly terrorist attack on innocent and defenseless worship
ers in egypt. the world cannot tolerate terrorism, we must defeat them milita
rily and discredit the extremist ideology that forms the basis of their exist
ence!

   horrible and cowardly terrorist attack on innocent and defenseless worship
ers in egypt. the world cannot tolerate terrorism, we must defeat them milita
rily and discredit the extremist ideology that forms the basis of their exist
ence!

   nyc terrorist was happy as he asked to hang isis flag in his hospital roo
m. he killed 8 people, badly injured 12. should get death penalty!

   nyc terrorist was happy as he asked to hang isis flag in his hospital roo
m. he killed 8 people, badly injured 12. should get death penalty!

   fake news cnn made a vicious and purposeful mistake yesterday. they were c
aught red handed, just like lonely brian ross at abc news (who should be imme
diately fired for his “mistake”). watch to see if @cnn fires those responsibl
e, or was it just gross incompetence?
```

In [175]:

```
print('Most positive tweets:')
for t in trump.sort_values('polarity', ascending=False).head()['text']:
    print('\n  ', t)
```

```
Most positive tweets:

   it was my great honor to celebrate the opening of two extraordinary museum
s-the mississippi state history museum &amp; the mississippi civil rights mus
eum. we pay solemn tribute to our heroes of the past &amp; dedicate ourselves
to building a future of freedom, equality, justice &amp; peace. https://t.co/
5akgvpv8aa

   it was my great honor to celebrate the opening of two extraordinary museum
s-the mississippi state history museum &amp; the mississippi civil rights mus
eum. we pay solemn tribute to our heroes of the past &amp; dedicate ourselves
to building a future of freedom, equality, justice &amp; peace. https://t.co/
5akgvpv8aa

   today, it was my great honor to sign a new executive order to ensure veter
ans have the resources they need as they transition back to civilian life. we
must ensure that our heroes are given the care and support they so richly des
erve! https://t.co/0mdp9ddias https://t.co/lp2a8kcbap

   today, it was my great honor to sign a new executive order to ensure veter
ans have the resources they need as they transition back to civilian life. we
must ensure that our heroes are given the care and support they so richly des
erve! https://t.co/0mdp9ddias https://t.co/lp2a8kcbap

   it was my great honor to welcome mayor’s from across america to the wh. my
administration will always support local government - and listen to the leade
rs who know their communities best. together, we will usher in a bold new era
of peace and prosperity! https://t.co/dmyectnk0a https://t.co/rsv7v7r0dt
```

## Question 6g

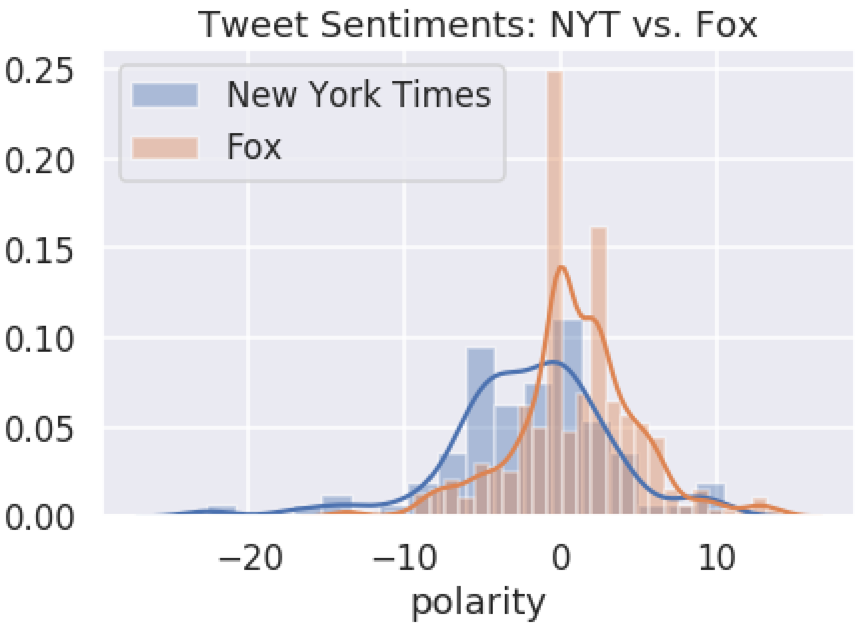
Plot the distribution of tweet sentiments broken down by whether the text of the tweet contains `nyt` or `fox`. Then in the box below comment on what we observe?

In [176]:

```
### BEGIN SOLUTION

nyt = sns.distplot(trump[trump['text'].str.contains("nyt")]['polarity'], label
= 'New York Times',)

fox = sns.distplot(trump[trump['text'].str.contains("fox")]['polarity'], label
= 'Fox')
plt.legend()

nyt.set_title('Tweet Sentiments: NYT vs. Fox')
```

Out[176]:

```
Text(0.5, 1.0, 'Tweet Sentiments: NYT vs. Fox')
```



**Comment on what you observe:**

The tweets that contain 'fox' generally have a higher polarity than tweets that contain 'nyt'. A higher polarity means that the tweet is more positive, while a negative polarity means the tweet is more negative.

# Question 7: Engagement

## Question 7a

In this problem, we'll explore which words led to a greater average number of retweets. For example, at the time of this writing, Donald Trump has two tweets that contain the word 'oakland' (tweets 932570628451954688 and 1016609920031117312) with 36757 and 10286 retweets respectively, for an average of 23,521.5.

Find the top 20 most retweeted words. Include only words that appear in at least 25 tweets. As usual, try to do this without any for loops. You can string together ~7 pandas commands and get everything done on one line.

Your `top_20` table should have this format:

| | retweet_count |
|---|---|
| **word** | |
| **jong** | 40675.666667 |
| **try** | 33937.800000 |
| **kim** | 32849.595745 |
| **un** | 32741.731707 |
| **maybe** | 30473.192308 |

Note that the contents of the table may be different based on how many tweets you pulled and when you did so; focus on the format, not the numbers.

In [177]:

```
### BEGIN SOLUTION


top_20 = (tidy_format
          .join(trump['retweet_count'])
          .loc[:, ['word', 'retweet_count']]
          .groupby('word')
          .filter(lambda temp: len(temp) > 24)
          .groupby('word')
          .median()
          .sort_values('retweet_count', ascending=False))

top_20.head()
```

Out[177]:

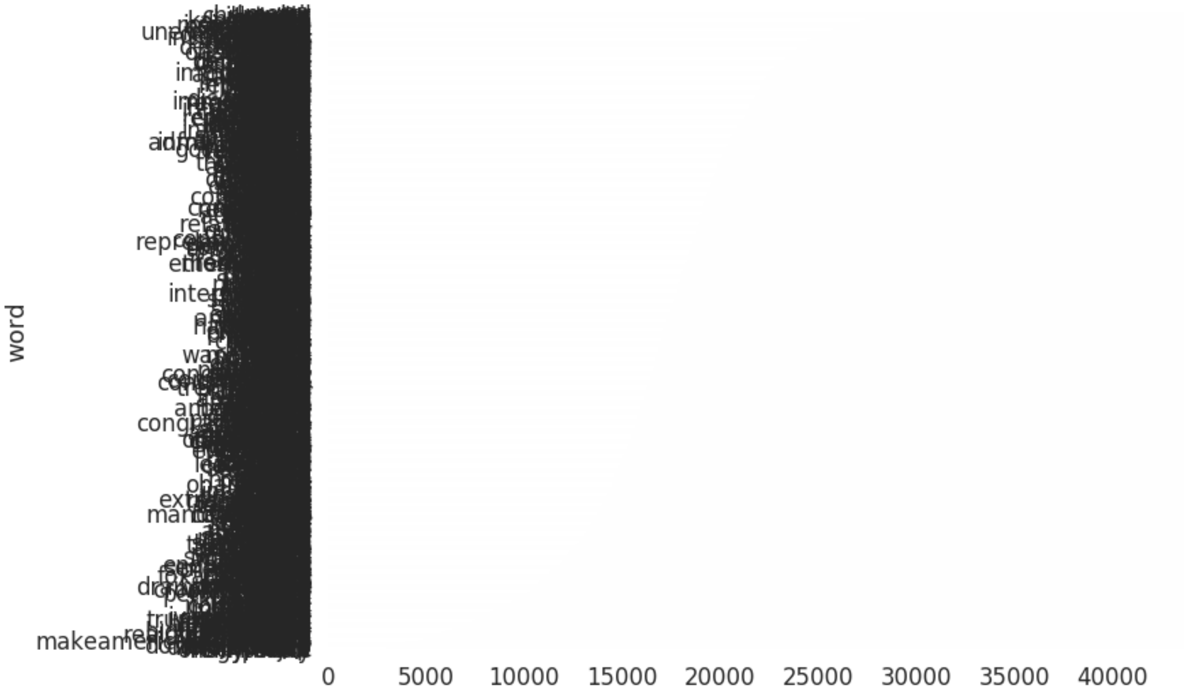
| word | retweet_count |
|---|---|
| merry | 41582.0 |
| jail | 35442.0 |
| christmas | 31870.0 |
| try | 31659.0 |
| illegally | 31586.0 |

In [178]:

```
# Although it can't be guaranteed, it's very likely that some of these words w
ill be in the top 20
# Although this may vary depending on when exactly you pulled your data:
assert 'un'      in top_20.index
assert 'nuclear' in top_20.index
assert 'old'     in top_20.index
assert 'nfl'     in top_20.index
```

Here's a bar chart of your results:

```
In [179]: top_20['retweet_count'].sort_values().plot.barh(figsize=(10, 8));
```



## Question 7b

At some point in time, "kim", "jong" and "un" were apparently really popular in Trump's tweets! It seems like we can conclude that his tweets involving jong are more popular than his other tweets. Or can we?

Consider each of the statements about possible confounding factors below. State whether each statement is true or false and explain. If the statement is true, state whether the confounding factor could have made kim jong un related tweets higher in the list than they should be.

1. We didn't restrict our word list to nouns, so we have unhelpful words like "let" and "any" in our result.
2. We didn't remove hashtags in our text, so we have duplicate words (eg. #great and great).
3. We didn't account for the fact that Trump's follower count has increased over time.

1. True. However, this will not cause "kim", "jong" and "un" to top the list of retweeted words since restricting to nouns does not affect the count of the retweets containing "kim", "jong" and "un".
2. False. We removed hashtags in our text when we removed punctuation.
3. True. This could indeed cause "kim", "jong" and "un" to appear higher on the list than it should have. If his follower count increased over time, we would expect the number of retweets over time to increase as well, regardless of what words are in the tweets. If he just started using the term "fake news" recently, it's likely that those tweets would get more retweets just because he had more followers than before.

# Question 8

Using the `trump` tweets construct an interesting plot describing a property of the data and discuss what you found below.

**Ideas:**

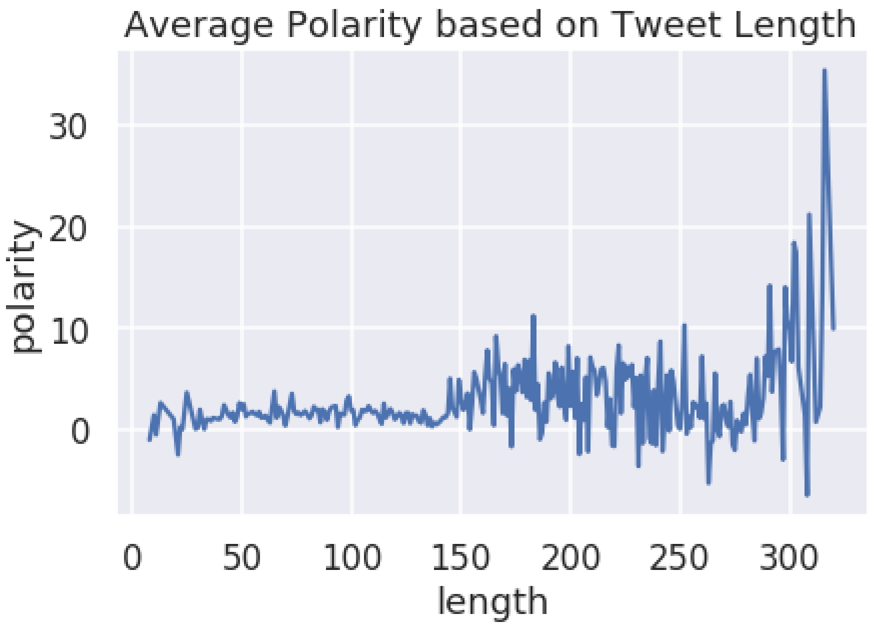
1. How has the sentiment changed with length of the tweets?
2. Does sentiment affect retweet count?
3. Are retweets more negative than regular tweets?
4. Are there any spikes in the number of retweets and do the correspond to world events?
5. *Bonus:* How many Russian twitter bots follow Trump?
6. What terms have an especially positive or negative sentiment?

You can look at other data sources and even tweets.

## Plot:

```
In [180]: #1. How has sentiment changed with length of the tweets?
          trump['length'] = [len(tweet) for tweet in trump['no_punc']]
          trump.head()
          temp = trump.groupby('length').mean()
          temp.head()
          temp = temp.drop(columns=['retweet_count', 'hour', 'year'])
          temp = temp.loc[temp['polarity'] != 0]
          plot = sns.lineplot(x=temp.index, y='polarity', data=temp)
          temp.tail()
          plot.set_title('Average Polarity based on Tweet Length')
```

```
Out[180]: Text(0.5, 1.0, 'Average Polarity based on Tweet Length')
```



### Discussion of Your Plot:

My lineplot shows the average polarity for each tweet based on the tweet length. As we can see from the plot, tweets from around 0-150 in length generally have positive polarities, ranging from about 0-4. However, as the lengths go past 150, the plot fluctuates but still remains mostly positive.

## Submission

Congrats, you just finished Project 1!

```
In [ ]:
```