News Mood

John M Warlop

Dependencies

In [1]:

```
import tweepy
import json
import pandas as pd
import requests
import pickle
```

Constants

In [2]:

```
handles = {'BBC':'@BBC', 'CBS':'@CBS', 'NYTimes':'@nytimes', 'Fox':'@
FoxNews', 'CNN':'@CNN'}
type_handles = ['compound','neu','neg','pos','dtime']
```

Functions

In [3]:

```
def get_keys():
    s_fname = 'api_keys'
    api_keys = {}
    with open(s_fname) as f:
        api_keys = json.load(f)
    f.close()
    return(api_keys)

def pickle_it(data,fname):
    import pickle
    pckl_out = open(fname,"wb")
    pickle.dump(data,pckl_out)
    pckl_out.close()

def logging_(log_name):
    import logging
    LOG_FORMAT = "%(levelname)s %(asctime)s - %(message)s"
    logging.basicConfig(filename=log_name,level=logging.INFO,format=L
OG_FORMAT,filemode='w')
    return(logging.getLogger())

def twitter_handshake(api_keys):
    auth = tweepy.OAuthHandler(api_keys['consumer_key'],api_keys['con
sumer_secret_key'])
    auth.set_access_token(api_keys['access_token'],api_keys['access_t
oken_secret'])
    api = tweepy.API(auth,parser=tweepy.parsers.JSONParser())
    return(api)

def sentiment_analysis(tweet):
    # Import and Initialize Sentiment Analyzer
    from vaderSentiment.vaderSentiment import SentimentIntensityAnaly
zer
    analyzer = SentimentIntensityAnalyzer()
    #results is dict{'compound':w,'neg':x,'neu':y,''pos':z} where w,
x,y and z are floats
    return(analyzer.polarity_scores(tweet))

def return_data_vector(data,name,type_):
    the_list = data[name]
    r_vector = []
    for idx, cell in enumerate(the_list):
        r_vector.append(cell[type_])
    return(r_vector)

def combine_vectors(data):
    all_vectors = {}
    for key,val in handles.items():
```

```
        all_vectors[key]={}
        for sentiment_param in type_handles:
            all_vectors[key][sentiment_param]=return_data_vector(data
,key,sentiment_param)
    return(all_vectors)
```

Solution Space

In [4]:

```
# Logging
logger = logging_("main.log")
```

In [5]:

```
# Twitter API Keys
logger.info("Loading API keys")
api_keys = get_keys()
logger.info("Keys loaded")
```

In [6]:

```
# Tweepy/Twitter Handshake
logger.info("Sending keys to twitter")
api = twitter_handshake(api_keys)
logger.info("Keys accepted by twitter")
```

Pickle Tweets

In [7]:

```
last_200_tweets = {}
for idx, key in enumerate(handles): #Loop through spewers of fake new
s('CNN','FOX', etc)
    public_tweets = api.user_timeline(handles[key],count=200,tweet_mo
de="extended")
    last_200_tweets[key]=public_tweets
pickle_it(last_200_tweets,"last_200_tweets.pkl")
del(last_200_tweets)
```

Open Pickle

In [8]:

```
pkl_in = open("last_200_tweets.pkl","rb")
last_200_tweets = pickle.load(pkl_in)
pkl_in.close()
```

Sentimate Analysis => analysis_store

In [9]:

```
analysis_store = {}
for idx0,fake_news in enumerate(last_200_tweets): #Loop through spewe
rs of fake news(CNN,FOX, etc)
    logger.info("Analyzing sentiment for {}".format(fake_news))
    analysis_store[fake_news] = []
    for idx1, e in enumerate(last_200_tweets[fake_news]):
        analysis = sentiment_analysis(last_200_tweets[fake_news][idx1
]['full_text'])
        tweet_dtime = last_200_tweets[fake_news][idx1]['created_at']
        analysis['dtime']=tweet_dtime
        analysis_store[fake_news].append(analysis)
        if idx1 % 20 == 0:
            logger.info("{} analysis is: {}".format(fake_news,analysi
s))
            logger.info("Analyzing tweet {} of {} from {}".\
                        format(idx1,len(last_200_tweets[fake_news]),f
ake_news))
        uname = e['user']['name']
```

Pickle Sentiment(save analysis_store)

In [10]:

```
#analysis_store['CNN'][55] retrieves 55th tweet
pickle_it(analysis_store,"sentiment_analysis.pkl")
```

Unpickle to analysis_store

In [11]:

```
pkl_in = open("sentiment_analysis.pkl","rb")
analysis_store = pickle.load(pkl_in)
pkl_in.close()
```

Do Sentiment Analysis

In [12]:

```
fake_news_vectors = combine_vectors(analysis_store)
#fake_news_vectors['CBS']['dtime']
```

Put Sentiment Analysis in DataFrame and Pickle(save df)

In [13]:

```
df_data = {}
for fake_news_name in fake_news_vectors:
    for type_ in type_handles:
        df_data[fake_news_name+'_'+type_] = fake_news_vectors[fake_ne
ws_name][type_]
df = pd.DataFrame(df_data)
pickle_it(df,"fake_news_sentiment_analysis_df.pkl")
```

Unpickle to DataFrame

In [14]:

```
import pickle
pkl_in = open("fake_news_sentiment_analysis_df.pkl","rb")
df = pickle.load(pkl_in)
pkl_in.close()
df = df.iloc[::-1] #flip table
```

Plot Compound Scores

In [15]:

```
import matplotlib.pyplot as plt
#import pandas as pd
xr = [i for i in range(-1*df.shape[0]+1,1)]
df['tago']=xr
df.head()
```

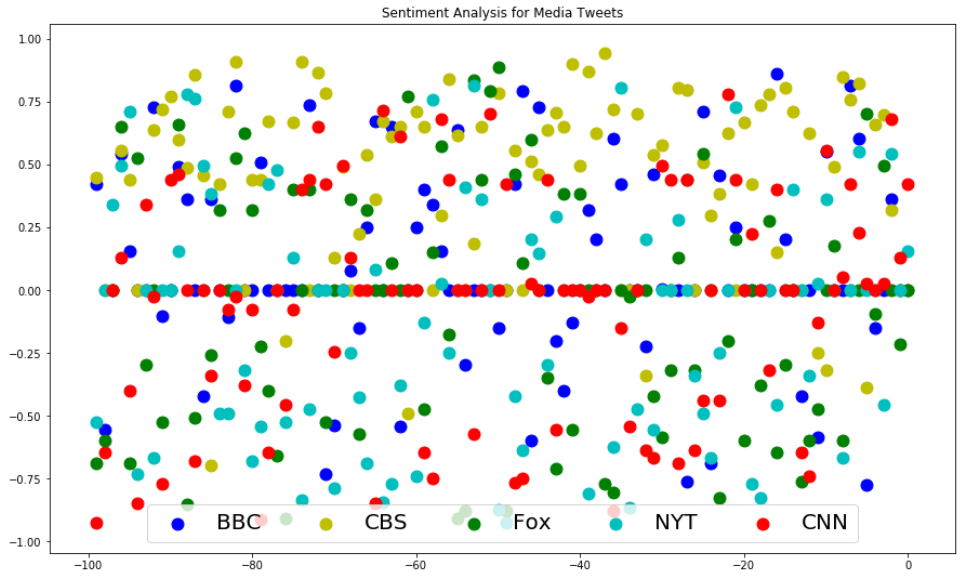
Out[15]:

	BBC_compound	BBC_dtime	BBC_neg	BBC_neu	BBC_pos	CBS_
199	0.0000	Wed Mar 28 13:32:04 +0000 2018	0.000	1.000	0.000	0.000
198	0.7506	Wed Mar 28 15:35:27 +0000 2018	0.000	0.720	0.280	0.585
197	0.0000	Wed Mar 28 16:00:10 +0000 2018	0.000	1.000	0.000	0.000
196	-0.8977	Wed Mar 28 16:30:00 +0000 2018	0.291	0.709	0.000	0.669
195	-0.3400	Wed Mar 28 17:00:12 +0000 2018	0.117	0.812	0.071	0.811

5 rows × 26 columns

In [16]:

```
plt.rcParams['figure.figsize']=(15,9)
#df.plot.scatter(x='tago',y='BBC',c=['g'],s=50)
#ax = df.plot(kind='scatter', x='tago', y='BBC', color='r')
#df.plot(kind='scatter', x='tago', y='CBS',color='g')
fig = plt.figure()
ax = fig.add_subplot(111)
sz=120
sz2 = 100 #Requirements call for last 100 tweets
ax.scatter(xr[100:], df['BBC_compound'].tolist()[100:], s=sz, c='b',
marker="o", label='BBC')
ax.scatter(xr[100:],df['CBS_compound'].tolist()[100:],s=sz,c='y',mark
er='o',label='CBS')
ax.scatter(xr[100:],df['Fox_compound'].tolist()[100:],s=sz,c='g',mark
er='o',label='Fox')
ax.scatter(xr[100:],df['NYTimes_compound'].tolist()[100:],s=sz,c='c',
marker='o',label='NYT')
ax.scatter(xr[100:],df['CNN_compound'].tolist()[100:],s=sz,c='r',mark
er='o',label='CNN')
plt.legend(loc='lower center',ncol=2);
ax.legend(frameon=True, loc='lower center', ncol=5, fontsize=20)
ax.set_title('Sentiment Analysis for Media Tweets')
plt.show()
```



Save Scatterplot

In [17]:

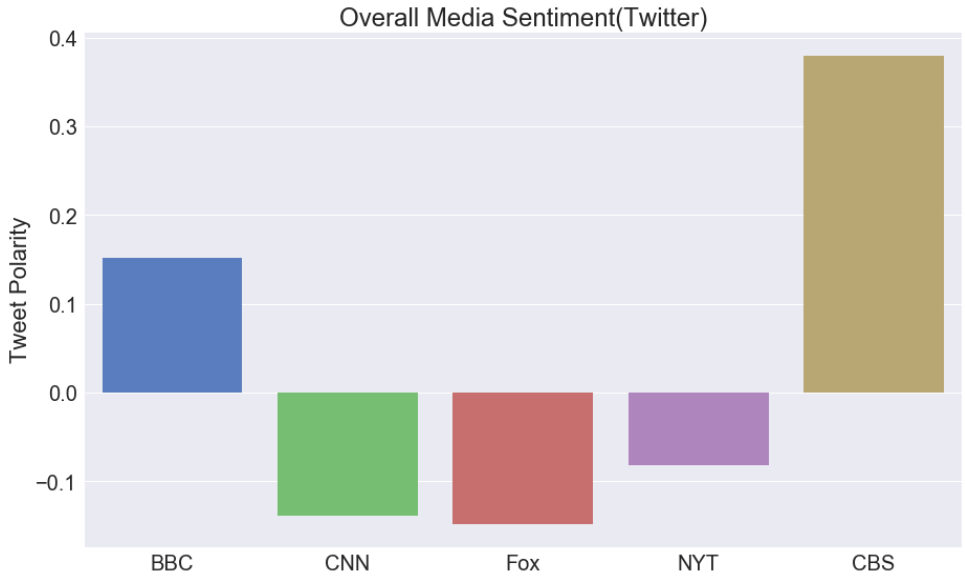
```
fig.savefig('sentiment_analysis_scatter.png')
```

Bar Chart (using Seaborn)

In [18]:

```
import seaborn as sns
bbc_mean = sum(df['BBC_compound'].tolist()[0:sz2])/sz2
cnn_mean = sum(df['CNN_compound'].tolist()[0:sz2])/sz2
fox_mean = sum(df['Fox_compound'].tolist()[0:sz2])/sz2
nyt_mean = sum(df['NYTimes_compound'].tolist()[0:sz2])/sz2
cbs_mean = sum(df['CBS_compound'].tolist()[0:sz2])/sz2
sns.set_style("darkgrid")
sns.set(font_scale=2)  # kinda big
bar_plot = sns.barplot(x=pd.Series(['BBC','CNN','Fox','NYT','CBS']),\
                       y=pd.Series([bbc_mean,cnn_mean,fox_mean,nyt_me
an,cbs_mean]),\
                       palette = "muted")
bar_plot.set(ylabel='Tweet Polarity',title='Overall Media Sentiment(T
witter)')

plt.xticks(rotation=0)
plt.show()
```



Save Seaborn BarPlot

In [19]:

```
fig = bar_plot.get_figure()
fig.savefig('sentiment_analysis_bar.png')
```

Save DataFrame to CSV

In [20]:

```
df.to_csv('sentiment_analysis.csv')
```