NLP - How Trump's Tweets impact China Market

The Motivation of the Project

This project is a basic investigation of the natural language process. Analyze how Trump's Tweets will impact China Market by checking what would happen in markets after Trump said positive, neutral or negative things about China. I get the idea from https://www.youtube.com/watch?v=wlnx-7cm4Gg&t=890s (https://www.youtube.com/watch?v=wlnx-7cm4Gg&t=890s)

A brief summary:

1. Get Trump's twitter data from Twitter Developer (a token is required, so you need to sign up an account at https://developer.twitter.com/en (https://developer.twitter.com/en))
2. Clean twitter data, including:

    - Extract main sentences by regularization
    - Find all root words (e.g. turn 'took' into 'take')
    - Delete stop words (e.g. deleting 'a', 'the')
    - Filter tweets with keywords: I use China and Chinese, only the tweets Have one of the keywords will be used in further research
    - Adjust time value: The time recode in twitter is neither US time nor China time. Turn it into China time, and if it is later than 15 p.m. (stock market close at this time), the date will lag for 1 day. (like, tonight's twitter will only influence tomorrow's market)
    - Analyzing the sentiment of a tweet (by third-party package). If there are more than 2 related tweets, the sentiment of that day would be weighted by retweets value.

3. Download Shanghai market data, by yahoo finance.
4. Analyze the correlation between two data:

    - correlation between sentiments and return of that day
    - set classes of sentiment and return, find the correlation between two class

Preliminary result

1. kind of no strong correlation.
2. An interesting thing is that I did similar steps in March (data from Dec to Mar), the correlation is strong than now.

Problems (need to develop):

1. The data set is small. A free account of twitter developer could fetch only 3200 tweets from a person, including replies. That's only 2 or 3 months data, cuz Trump used twitter tooooo much.
2. The tweet on weekends was not considered. (there might be lots of news could impact markets at weekends)
3. The analyzing part is simple and superficial.
4. Using daily data now, minutes data would be much better.
5. Sentiment analysis is given by a third-party package, which is not precise, sometimes.

Possible Future Work

1. I personally find it is an interesting NLP project, and the idea could used to other places, like how CNN's news impact market.
2. The key point is how to combine markets and tweets, by quantitative methods. Otherwise, it is completely a data science project.

In [ ]:
```
from tweepy import API
from tweepy import Cursor
from tweepy.streaming import StreamListener
from tweepy import OAuthHandler
from tweepy import Stream

import datetime
import numpy as np
import pandas as pd
import nltk
from textblob import TextBlob
import matplotlib.pyplot as plt
import re
from pandas_datareader import data as pdr
import yfinance as yf
from scipy.stats import f


# Variables that contains the user credentials to access Twitter API
# You can get key and token from https://developer.twitter.com/en
CONSUMER_KEY    =
CONSUMER_SECRET =
# Access:
ACCESS_TOKEN  =
ACCESS_TOKEN_SECRET =


# # # # TWITTER CLIENT # # # #
class TwitterClient():
    def __init__(self, twitter_user=None):
        self.auth = TwitterAuthenticator().authenticate_twitter_app()
        self.twitter_client = API(self.auth)
        self.twitter_user = twitter_user

    def get_twitter_client_api(self):
        return self.twitter_client


# # # # TWITTER AUTHENTICATER # # # #
class TwitterAuthenticator():
    def authenticate_twitter_app(self):
        auth = OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
        auth.set_access_token(ACCESS_TOKEN, ACCESS_TOKEN_SECRET)
        return auth


# # # # TWITTER STREAMER # # # #
class TwitterStreamer():
# monitor the live twitter, return json
    def __init__(self):
        self.twitter_autenticator = TwitterAuthenticator()

    def stream_tweets(self, fetched_tweets_filename, hash_tag_list):
        # This handles Twitter authetification and the connection to Twitter Stream
ing API
        listener = TwitterListener(fetched_tweets_filename)
        auth = self.twitter_autenticator.authenticate_twitter_app()
        stream = Stream(auth, listener)
        # This line filter Twitter Streams to capture data by the keywords:
        stream.filter(track=hash_tag_list)


# # # # TWITTER STREAM LISTENER # # # #
```

```
class TwitterListener(StreamListener):
    def __init__(self, fetched_tweets_filename):
        self.fetched_tweets_filename = fetched_tweets_filename

    def on_data(self, data):
        try:
            print(data)
            with open(self.fetched_tweets_filename, 'a') as tf:
                tf.write(data)
            return True
        except BaseException as e:
            print("Error on_data %s" % str(e))
        return True

    def on_error(self, status):
        if status == 420:
            # Returning False on_data method in case rate limit occurs.
            return False
        print(status)


# # # # TWITTER ANALYSIS # # # #
class TweetAnalyzer():
### process the historical twitter
    def tweets_to_data_frame(self, tweets):
        df = pd.DataFrame(data=[tweet.text for tweet in tweets], columns=['Tweets
'])
        df['id'] = np.array([tweet.id for tweet in tweets])
        df['len'] = np.array([len(tweet.text) for tweet in tweets])
        df['date'] = np.array([tweet.created_at for tweet in tweets])
        df['source'] = np.array([tweet.source for tweet in tweets])
        df['likes'] = np.array([tweet.favorite_count for tweet in tweets])
        df['retweets'] = np.array([tweet.retweet_count for tweet in tweets])
        df['Tweets'] = df.Tweets.apply(self.clean_tweet)
        return df

    # catch meaning for part, clean stop word, find root
    def clean_tweet(self, tweet):
        stop_words = nltk.corpus.stopwords.words('english')
        lemmatizer = nltk.stem.WordNetLemmatizer()
        stemmer = nltk.stem.PorterStemmer()
        words = re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)", " ", twee
t).split()
        # kick out stop words
        words = [w for w in words if w not in stop_words]
        # word root
        words = [lemmatizer.lemmatize(word) for word in words]
        # words = [stemmer.stem(word) for word in words]
        # return ' '.join(words)
        return words

    # find key word in twitter, return 0 or 1
    def find_key_word(self, tweet):
        for key_word in key_word_list:
            if key_word in tweet:
                return 1
        return 0

    # return sentiment, need to input a whole string
    def analyze_sentiment(self, tweet):
        analysis = TextBlob(" ".join(tweet))
        return analysis.sentiment.polarity
```

In [2]:
```
#fetch live twitter
#hash_tag_list = ["donal trump", "hillary clinton", "barack obama", "bernie sander
s"]
#fetched_tweets_filename = "tweets.txt"
#twitter_streamer = TwitterStreamer()
#twitter_streamer.stream_tweets(fetched_tweets_filename, hash_tag_list)
```

In [6]:
```
# download twitter data
def fetch_twitter(name = "realDonaldTrump", pages = 16, key_word_list = ['China','C
hinese']):
    print('Downloading Tweets...\t')
    twitter_client = TwitterClient()
    tweet_analyzer = TweetAnalyzer()
    api = twitter_client.get_twitter_client_api()

    # can only get 200*16 as maximum
    tweets = api.user_timeline(screen_name = name, count=200, page=1)  #exclude_rep
lies = True, include_rts=False
    df = tweet_analyzer.tweets_to_data_frame(tweets)
    for i in range(2, pages+1):
        tweets = api.user_timeline(screen_name = name, count=200, page=i)
        df = pd.concat([df,tweet_analyzer.tweets_to_data_frame(tweets)])
    df = df.reset_index()

    # sentiment analysis
    df['sentiment'] = df['Tweets'].apply(tweet_analyzer.analyze_sentiment)
    # find the tweets with key words
    df['include_keyword'] = df['Tweets'].apply(tweet_analyzer.find_key_word)
    print('Total Tweets:',len(df),' tweets with key words:',sum(df.include_keywor
d))
    # turn time into China time
    df['effect_date'] = pd.to_datetime(df.date).apply(lambda x:(x + datetime.timede
lta(hours=20)))
    # lag for one day if the time is latter than 15 pm
    df['effect_date'] = df.effect_date.apply(lambda x:(x+datetime.timedelta(days=1)
if x.hour>15 else x).strftime("%Y-%m-%d"))

    return df.loc[df.include_keyword==1,['Tweets','len','date','effect_date','likes
','retweets','sentiment']]


# download market data
def fetch_stock_data(df, ret_upper = 0.003, ret_lower = -0.003):
    print('Downloading Stock Data...\t')
    datelist = sorted(list(df.effect_date.unique()))
    yf.pdr_override()
    ss_data = pd.DataFrame(pdr.get_data_yahoo('000001.SS', start=datelist[0], end=d
atelist[-1])['Adj Close'])
    ss_data['Return'] = ss_data['Adj Close']/ss_data['Adj Close'].shift(1) - 1
    ss_data['Return_Class'] = ss_data.Return.apply(lambda x:1 if x > ret_upper else
-1 if x < ret_lower else 0)
    ss_data.index = ss_data.index.strftime("%Y-%m-%d")
    ss_data = ss_data.dropna()

    return ss_data.loc[ss_data.index.isin(datelist)]


def process(sen_upper = 0.1, sen_lower = 0.1):
    df = fetch_twitter(name = "realDonaldTrump", pages = 20, key_word_list = ['Chin
a','Chinese'])
    ret = fetch_stock_data(df, ret_upper = 0.005, ret_lower = -0.005)
    ret['Tweets_Amount'], ret['Weighted_Sentiment'], ret['Sentiment_Class'] = 0,0,0
    for date in ret.index:
        tem = df.loc[df.effect_date == date]
        ret.loc[ret.index == date,'Tweets_Amount'] = len(tem)
        # sentiment is weighted by retweets value
        tem_sentiment = sum(tem['retweets'] * tem['sentiment'])/sum(tem['retweets
'])
        ret.loc[ret.index == date,'Weighted_Sentiment'] = tem_sentiment
        if tem_sentiment >= 0.05:
```

```
            ret.loc[ret.index == date,'Sentiment_Class'] = ret.loc[ret.index == dat
e,'Weighted_Sentiment'].apply(\
            lambda x:1 if x > sen_upper else -1 if x < sen_lower else 0)

    return df,ret
```

In [7]:
```
if __name__ is '__main__':
    key_word_list = ['China','Chinese']
    twitter_data, ret = process()
    print(np.corrcoef(ret.Return,ret.Weighted_Sentiment))
    print(np.corrcoef(ret.Return_Class,ret.Sentiment_Class))
```

```
Downloading Tweets...
Total Tweets: 3234  tweets with key words: 64
Downloading Stock Data...
[**********************100%**********************]  1 of 1 completed
[[ 1.         -0.13029578]
 [-0.13029578  1.        ]]
[[1.         0.17073699]
 [0.17073699 1.        ]]
```

In [13]:
```
twitter_data.head()
```

Out[13]:

	Tweets	len	date	effect_date	likes	retweets	sentiment
152	[Biden, failed, China, They, took, u, cleaner,...	129.0	2020-06-24 22:10:48	2020-06-26	164723.0	34804.0	-0.150000
175	[We, great, job, CoronaVirus, including, early...	140.0	2020-06-23 14:41:50	2020-06-24	122105.0	25251.0	0.450000
189	[The, China, Trade, Deal, fully, intact, Hopef...	108.0	2020-06-23 02:22:18	2020-06-24	121205.0	22762.0	0.136364
438	[How, think, China, Russia, Japan, others, wou...	140.0	2020-06-12 12:37:47	2020-06-13	75432.0	19059.0	0.000000
494	[RT, Decades, failed, engagement, Western, inv...	140.0	2020-06-11 03:30:48	2020-06-12	0.0	11639.0	-0.333333

In [14]:
```
ret.head()
```

Out[14]:

	Adj Close	Return	Return_Class	Tweets_Amount	Weighted_Sentiment
Date					
2020-03-26	2764.910889	-0.005997	-1	2	0.015969
2020-04-08	2815.368896	-0.001912	0	1	0.138095
2020-04-09	2825.904053	0.003742	0	1	0.000000
2020-04-13	2783.048096	-0.004857	0	1	-0.083333
2020-04-14	2827.282959	0.015894	1	2	-0.286314