# twitter-stream-sentiment-analysis

November 11, 2019

## 1 Twitter Sentiment Analysis using Spark Streams

The case study seeks to read incoming tweets and process them.

### 1.1 Installation

Modern Data sources typically deliver their data over an API. In this exercise, we will first acquire some tweets and work with them in various ways.

We'll be working with `tweepy`, an API for acquiring Data from Twitter.The best place to start is the documentation. Take a look at the Getting Started page.

To begin, install `tweepy` using pip. You need only do it once (but doing it more than once won't hurt anything). To install it, run the next cell.

```
[ ]: !pip install tweepy
```

The next step is to obtain Twitter credentials for being able to acccess the API.Visit the Twitter Developer Site for Apps to create a twitter "application," give it a name.

Once created, go to the keys and tokens tab and copy-paste the Consumer API keys as well as the Access token & access token secret into the cell below.

Careful! `consumer_secret` and `access_token_secret` should be protected like passwords because they can be used by the API to *send out tweets or direct messages* on your behalf. Immediately invalidate the token if you accidentally expose it — regenerating a new set of token values on the keys and tokens page will invalidate the old one.

To protect Twitter credentials from being made public I prefer to store them in Google Storage and retrieve them on the fly using the Cloud Storage Client Libraries.

A file `twitter_creds.json` stores the credentials in json format.

```
{
    consumer_key =        '(copied from twitter dev)',
    consumer_secret =     '(---- copied ---- from ---- twitter ---- dev ----)',
    access_token =        '(---- copied ---- from ---- twitter ---- dev ----)',
    access_token_secret = '(--- copied --- from --- twitter --- dev ---)'
}
```

Alternatively, you could store the credentials in environment variables on your machine and retrieve them using os.environ. However, if you're accessing the environment variable from a Jupyter notebook, I've found that the environment variable are sometimes not available to an iPython session (depending on how your Jupyter environment was set up) so you may have to experiment a little bit to get it to work. My work session is shown below.

```
[ ]: !pip install google-cloud-storage
```

```
[1]: %set_env GOOGLE_APPLICATION_CREDENTIALS=/home/singhj/.ssh/.google_credentials.
      ↪json
```

```
env: GOOGLE_APPLICATION_CREDENTIALS=/home/singhj/.ssh/.google_credentials.json
```

```
[ ]: import json
     from google.cloud import storage
     storage_client = storage.Client()
     bucket = storage_client.get_bucket("python-at-tufts-2019")
     blob = storage.blob.Blob("twitter-creds.json", bucket)
     creds_bytes = blob.download_as_string()
     _creds = json.loads(str(creds_bytes, 'utf-8'))

     import tweepy
     auth = tweepy.OAuthHandler(_creds["consumer_key"], _creds["consumer_secret"])
     auth.set_access_token(_creds["access_token"], _creds["access_token_secret"])
     del _creds    # delete _creds immediately!

     API = tweepy.API(auth, wait_on_rate_limit=True, wait_on_rate_limit_notify=True)
```

```
[ ]: def show_tweet(tweet):
         try:
             return ' '.join([tweet.created_at.strftime("%m/%d/%Y, %H:%M:%S"),
                                  tweet.user.screen_name,
                                  tweet.text])
         except:
             return '************************ bad or no tweet␣
      ↪************************'
```

```
[4]: public_tweets = API.home_timeline()
     for tweet in public_tweets[:1]:
         print(show_tweet(tweet))
```

```
11/11/2019, 20:29:31 brianlaungaoaeh RT @JamesBakerCI: After overseeing the
Hamburg Süd acquisition, Maersk chief operating officer Søren Toft is going to
pursue outside opport…
```

## 1.2 Did you get some tweets?

Good! The API documentation is here. We will use the API for finding information about your favorite Twitter user: things like their latest tweets, their followers, ...

[ ]:
```
# First enter the name of a prominent user, say @nytimes.
nyt = API.get_user('@nytimes')
```

[ ]:
```
# Tweets from your prominent user
user_tweets = API.user_timeline(user)
for tweet in user_tweets[:3]:
    print(show_tweet(tweet))
```

[ ]:
```
# Some followers of your favorite user
user_friends = API.friends(user)
# print ([friend.screen_name for friend in user_friends])
tweeters = [nyt.id_str] + [friend.id_str for friend in user_friends]
# tweeters
```

## 2 TO-DO #1

Add @nytimes's followers to `tweeters` until you have 5000 distinct tweeters. [3½ points for reproducing the above steps in a pyspark environment, creating the list of 5000 distinct tweeters and dealing with any rate limits imposed by Twitter.]

Running the next cell starts the stream. In a Jupyter environment, you'll need to interrupt the kernel to stop it.

[ ]:
```
import sys

class Listener(tweepy.streaming.StreamListener):
    def __init__(self, output_file=sys.stdout):
        super(Listener,self).__init__()
        self.output_file = output_file
    def on_status(self, tweet):
        print(show_tweet(tweet), file=self.output_file)
    def on_error(self, status_code):
        print(status_code)
        return False

listener = Listener()

stream = tweepy.Stream(auth=API.auth, listener=listener)
print ('Starting!!!!!!!!!!!!!!!!!!')
#stream.filter(follow=[nyt.id_str])
try:
    print('Start streaming.')
    stream.filter(follow=tweeters)
    #stream.sample(languages=['en'])
except KeyboardInterrupt:
    print("Stopped.")
finally:
    stream.disconnect()
    print('Done.')
```

## 3 TO-DO #2

1. Download AFINN Data and convert it to a Python Dictionary `sentiment_dict`. [no points for this]
2. Accumulate the sentiment score for each word in the tweet text; if a word is not in `sentiment_dict` its weight should be considered zero. [2 points]
3. Using a sliding window of 1 minute and a sliding interval of 5 seconds, create a new DStream `'summary'` that includes [3 points]:
   - The number of tweets in that sliding window,
   - The average sentiment in that window,
   - The most popular hashtag and the average sentiment of all tweets with that hashtag in that window.
4. From the DStream `'summary'`, construct another DStream `'alerts'` that produces `(hashtag, sign, percentage, timestamp)` tuples whenever the sentiment of a particular hashtag over the prior 10 minutes changes by more than 25% in either direction. `sign` is to be the character '+' or '-' depending on the direction of the change [4 points].

## 4 TO-DO #3 [4 bonus points]

Twitter returns locations in terms of WOEIDs (where on earth ID).

WOEIDs can be interpreted using the WOEID package available through PyPi. To operate, the package requires a Yahoo! application which has been deprecated. The bonus points are for finding a method for expressing WOEIDs as geolocation coordinates so they could be used on a map. For instance, I'm interested in mapping sentiment scores by region on the *speculation* that different parts of the country might express different sentiments about the same events.

[ ]:
```

```