# Twitter Analytics

May 7, 2019

## 1 Step one

Collect one lakh tweets on a specific domain name and duration as input

**Example**

**input** sports

| serial_number | screen_name | user_id | tweet_id | retweet_count | date | tweet |
|---|---|---|---|---|---|---|
| 1 | lorem | ip | 12 | 1 | 23 | xh |

**output format:** we'll be using `twitterscraper` for this purpose.

```
In [1]: %%bash
        twitterscraper python --limit 1000 --lang en --output ~/backups/today\'stweets.json
```

```
INFO: queries: ['python since:2006-03-21 until:2006-11-12', 'python since:2006-11-12 until:200
INFO: Querying python since:2006-03-21 until:2006-11-12
INFO: Querying python since:2006-11-12 until:2007-07-06
INFO: Querying python since:2007-07-06 until:2008-02-27
INFO: Querying python since:2008-02-27 until:2008-10-20
INFO: Querying python since:2008-10-20 until:2009-06-13
INFO: Querying python since:2009-06-13 until:2010-02-04
INFO: Querying python since:2010-02-04 until:2010-09-29
INFO: Querying python since:2011-05-23 until:2012-01-14
INFO: Querying python since:2010-09-29 until:2011-05-23
INFO: Querying python since:2012-01-14 until:2012-09-06
INFO: Querying python since:2012-09-06 until:2013-04-30
INFO: Querying python since:2013-04-30 until:2013-12-22
INFO: Querying python since:2013-12-22 until:2014-08-15
INFO: Querying python since:2014-08-15 until:2015-04-09
INFO: Querying python since:2015-12-01 until:2016-07-24
INFO: Querying python since:2015-04-09 until:2015-12-01
INFO: Querying python since:2017-11-08 until:2018-07-02
INFO: Querying python since:2017-03-17 until:2017-11-08
```

```
INFO: Querying python since:2016-07-24 until:2017-03-17
INFO: Querying python since:2018-07-02 until:2019-02-24
INFO: Got 5 tweets for python%20since%3A2006-03-21%20until%3A2006-11-12.
INFO: Got 5 tweets (5 new).
INFO: Got 60 tweets for python%20since%3A2008-10-20%20until%3A2009-06-13.
INFO: Got 65 tweets (60 new).
INFO: Got 54 tweets for python%20since%3A2015-12-01%20until%3A2016-07-24.
INFO: Got 119 tweets (54 new).
INFO: Got 60 tweets for python%20since%3A2017-03-17%20until%3A2017-11-08.
INFO: Got 179 tweets (60 new).
INFO: Got 60 tweets for python%20since%3A2010-02-04%20until%3A2010-09-29.
INFO: Got 239 tweets (60 new).
INFO: Got 60 tweets for python%20since%3A2013-12-22%20until%3A2014-08-15.
INFO: Got 299 tweets (60 new).
INFO: Got 60 tweets for python%20since%3A2010-09-29%20until%3A2011-05-23.
INFO: Got 359 tweets (60 new).
INFO: Got 60 tweets for python%20since%3A2014-08-15%20until%3A2015-04-09.
INFO: Got 419 tweets (60 new).
INFO: Got 60 tweets for python%20since%3A2006-11-12%20until%3A2007-07-06.
INFO: Got 479 tweets (60 new).
INFO: Got 60 tweets for python%20since%3A2015-04-09%20until%3A2015-12-01.
INFO: Got 539 tweets (60 new).
INFO: Got 60 tweets for python%20since%3A2009-06-13%20until%3A2010-02-04.
INFO: Got 599 tweets (60 new).
INFO: Got 60 tweets for python%20since%3A2007-07-06%20until%3A2008-02-27.
INFO: Got 659 tweets (60 new).
INFO: Got 60 tweets for python%20since%3A2018-07-02%20until%3A2019-02-24.
INFO: Got 719 tweets (60 new).
INFO: Got 60 tweets for python%20since%3A2011-05-23%20until%3A2012-01-14.
INFO: Got 779 tweets (60 new).
INFO: Got 60 tweets for python%20since%3A2016-07-24%20until%3A2017-03-17.
INFO: Got 839 tweets (60 new).
INFO: Got 60 tweets for python%20since%3A2013-04-30%20until%3A2013-12-22.
INFO: Got 899 tweets (60 new).
INFO: Got 60 tweets for python%20since%3A2017-11-08%20until%3A2018-07-02.
INFO: Got 959 tweets (60 new).
INFO: Got 60 tweets for python%20since%3A2012-09-06%20until%3A2013-04-30.
INFO: Got 1019 tweets (60 new).
INFO: Got 60 tweets for python%20since%3A2012-01-14%20until%3A2012-09-06.
INFO: Got 1079 tweets (60 new).
INFO: Got 60 tweets for python%20since%3A2008-02-27%20until%3A2008-10-20.
INFO: Got 1139 tweets (60 new).
```

```
In [21]: import codecs
         import json
         import pandas as pd
         pd.options.mode.chained_assignment = None
```

```
from typing import List, Dict

def load_json_file(file_path: str) -> Dict:
    with codecs.open(file_path, "r", "utf-8") as f:
        return json.load(f, encoding="utf-8")

tweets = load_json_file("/home/vinay/backups/today\'stweets.json")

list_tweets = [list(elem.values()) for elem in tweets]
list_columns = list(tweets[0].keys())

twitter_data = pd.DataFrame(list_tweets, columns=list_columns)
twitter_data.head()
```

Out[21]:

| | timestamp | url | text | user | html | retweets | replies | fullname | id | likes |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2006-11-08T11:46:29 | /larskflem/status/59306 | coding python. happy time | larskflem | <p class="TweetTextSize js-tweet-text tweet-te... | 0 | 0 | Lars K. Flem | 59306 | 0 |
| 1 | 2006-11-06T21:20:39 | /sergio_101/status/57683 | Trying to figure out what phone to get next.. ... | sergio_101 | <p class="TweetTextSize js-tweet-text tweet-te... | 0 | 0 | sergio t. ruiz | 57683 | 0 |
| 2 | 2006-10-23T00:21:20 | /thomasknoll/status/46836 | Learning python while kim watches city of god | thomasknoll | <p class="TweetTextSize js-tweet-text tweet-te... | 0 | 0 | Thomas Knoll | 46836 | 0 |
| 3 | 2006-08-02T02:07:24 | /marceloeduardo/status/15613 | Finishing some turbogears experience, writing ... | marceloeduardo | <p class="TweetTextSize js-tweet-text tweet-te... | 1 | 0 | Marcelo Eduardo | 15613 | 1 |
| 4 | 2006-07-16T18:03:45 | /nitin/status/10584 | Heading to peets in emryvil to hack python tnx... | nitin | <p class="TweetTextSize js-tweet-text tweet-te... | 1 | 1 | Nitin Borwankar | 10584 | 1 |

We can drop columns `html`, `url`, `likes`, `replies`.

We need to modify `timestamp` column, add `user` and `fullname` columns. and get user_ids of the `user`.

order the columns, based on the given output format

```
In [22]: # making timestamp YYYY-MM-DD
         twitter_data['timestamp'] = twitter_data['timestamp'].apply(lambda x: x.split('T')[0])
```

```
        # dropping html, url, likes and replies
        twitter_data.drop(columns=['html', 'url', 'likes', 'replies'], inplace=True)

        # twitter_data.head()
        twitter_data.columns
```

```
Out[22]: Index(['timestamp', 'text', 'user', 'retweets', 'fullname', 'id'], dtype='object')
```

```
In [23]: # renaming column names
         twitter_data.columns = ['Date', 'Tweet', 'user', 'retweets', 'fullname', 'Tweet_id']

         twitter_data.head()
         twitter_data_backup = twitter_data
```

### 1.1 Step 2

from the step 1 output observe( 5th column of the table) i.e number of re tweets obtained for each tweet . If number of re tweets obtained for the given tweet is 0 then discard the tweet other wise print the tweet in the above format.

Output : print only the tweets which got re tweets and discard the tweets with no re tweets
This will contain the tweets with more than zero retweets.

```
In [4]: twitter_data = twitter_data[twitter_data.retweets != "0"]
        twitter_data.head()
```

```
Out[4]:          Date                                              Tweet  \
        3   2006-08-02  Finishing some turbogears experience, writing ...
        4   2006-07-16  Heading to peets in emryvil to hack python tnx...
        66  2016-07-23  tethne 0.8.1.dev8: Bibliographic network and c...
        71  2016-07-23  Thank @mandarlimaye 4 your follow and welcom #...
        72  2016-07-23  Thank @h1ng 4 your follow and welcom #PostgreS...

                    user retweets          fullname            Tweet_id
        3    marceloeduardo        1    Marcelo Eduardo               15613
        4             nitin        1    Nitin Borwankar               10584
        66  mastercodeonlin        1  MasterCode.Online  757001950088957952
        71       lennincaro        3        Lennin Caro  757000314071478272
        72       lennincaro        4        Lennin Caro  757000213622030336
```

### 1.2 for step three

Step 3: Find out number of users who has been tweeted those tweets in step 2, because one user may post multiple tweets.

Input: output of step 2
Output:

| serial_numberuser_name @mention | user_id | tweets (no of tweets posted by user) |
|---|---|---|

```
In [5]: # for step 3 date column is irrelevant
        # remove first date column
        twitter_data_with_date = twitter_data
        twitter_data.drop(columns=['Date', 'Tweet'], inplace=True)
        twitter_data.head()
```

Out[5]:

| | user | retweets | fullname | Tweet_id |
|---|---|---|---|---|
| 3 | marceloeduardo | 1 | Marcelo Eduardo | 15613 |
| 4 | nitin | 1 | Nitin Borwankar | 10584 |
| 66 | mastercodeonlin | 1 | MasterCode.Online | 757001950088957952 |
| 71 | lennincaro | 3 | Lennin Caro | 757000314071478272 |
| 72 | lennincaro | 4 | Lennin Caro | 757000213622030336 |

```
In [6]: # rather than dropping duplicated we can `groupby` in pandas
        # twitter_data.duplicated(subset='user', keep='first').sum()
        tweet_count = twitter_data.groupby(twitter_data.user.tolist(),as_index=False).size()
        tweet_count['mastercodeonlin']
```

Out[6]: 2

```
In [7]: def get_tweet_count(user: str) -> int:
            return tweet_count[user]

        get_tweet_count('mastercodeonlin')
```

Out[7]: 2

```
In [8]: twitter_data['no_of_tweets'] = twitter_data['user'].apply(lambda x: get_tweet_count(x))

        twitter_data_without_tweet_count = twitter_data.drop_duplicates(subset='user', keep="fi
        twitter_data_without_tweet_count.reset_index(drop=True, inplace=True)
        twitter_data_without_tweet_count.head()
```

Out[8]:

| | user | retweets | fullname | Tweet_id | \ |
|---|---|---|---|---|---|
| 0 | marceloeduardo | 1 | Marcelo Eduardo | 15613 | |
| 1 | nitin | 1 | Nitin Borwankar | 10584 | |
| 2 | mastercodeonlin | 1 | MasterCode.Online | 757001950088957952 | |
| 3 | lennincaro | 3 | Lennin Caro | 757000314071478272 | |
| 4 | devbattles | 9 | Dev Battles | 756996796786900993 | |

| | no_of_tweets |
|---|---|
| 0 | 1 |
| 1 | 1 |
| 2 | 2 |
| 3 | 5 |
| 4 | 2 |

In [9]:
```
# in order to get user_id for a user
# we need to use tweepy, need to work on getting user_ids twitterscraper way.

import tweepy

configs = load_json_file("configs.json")

APP_KEY = configs['APP_KEY']
APP_SECRET = configs['APP_SECRET']

# authenticate api
auth = tweepy.AppAuthHandler(APP_KEY, APP_SECRET)
api = tweepy.API(auth, wait_on_rate_limit=True, wait_on_rate_limit_notify=True)

if (not api):
    print("Can't Authenticate")
    sys.exit(-1)
```

In [10]:
```
# get user_id from screen name
def get_user_id_from_screen_name(screen_name: str, api: object) -> int:
    try:
        id = api.get_user(screen_name=screen_name).id
#         print(id)
        return id
    except tweepy.TweepError:
        return None

get_user_id_from_screen_name("nitin", api)
```

Out[10]: 988

In [11]:
```
twitter_data_without_tweet_count['user_id'] = twitter_data_without_tweet_count['user'
```

In [12]:
```
twitter_data_without_tweet_count.head()
```

Out[12]:

| | user | retweets | fullname | Tweet_id | no_of_tweets | user_id |
|---|---|---|---|---|---|---|
| 0 | marceloeduardo | 1 | Marcelo Eduardo | 15613 | 1 | 3652 |
| 1 | nitin | 1 | Nitin Borwankar | 10584 | 1 | 988 |
| 2 | mastercodeonlin | 1 | MasterCode.Online | 757001950088957952 | 2 | 3041232857 |
| 3 | lennincaro | 3 | Lennin Caro | 757000314071478272 | 5 | 205824307 |
| 4 | devbattles | 9 | Dev Battles | 756996796786900993 | 2 | 2377678050 |

### 1.3 for step four

All the users who are there in the output of step 3 are not influential users, to find out Influential users from the above table, find out no of retweets obtained for each user and calculate weight or user rank.

output format:

| serial_number | user_name@mention | user_id | #tweets (no of tweets posted by user) | # retweets | log(#retweets) |
|---|---|---|---|---|---|
| | | | | | |

```
In [13]: import math
         twitter_data_without_tweet_count['log(retweets)'] = twitter_data_without_tweet_count[
         twitter_data_without_tweet_count.head()
```

```
Out[13]:              user retweets          fullname            Tweet_id  \
         0   marceloeduardo        1    Marcelo Eduardo               15613
         1           nitin        1    Nitin Borwankar               10584
         2  mastercodeonlin        1  MasterCode.Online  757001950088957952
         3       lennincaro        3        Lennin Caro  757000314071478272
         4       devbattles        9        Dev Battles  756996796786900993

            no_of_tweets     user_id  log(retweets)
         0             1        3652       0.000000
         1             1         988       0.000000
         2             2  3041232857       0.000000
         3             5   205824307       1.098612
         4             2  2377678050       2.197225
```

```
In [14]: tw_data = twitter_data_without_tweet_count[['user', 'fullname', 'user_id', 'no_of_twe
         tw_data.head()
```

```
Out[14]:              user           fullname     user_id  no_of_tweets retweets  \
         0   marceloeduardo    Marcelo Eduardo        3652             1        1
         1           nitin    Nitin Borwankar         988             1        1
         2  mastercodeonlin  MasterCode.Online  3041232857             2        1
         3       lennincaro        Lennin Caro   205824307             5        3
         4       devbattles        Dev Battles  2377678050             2        9

            log(retweets)
         0       0.000000
         1       0.000000
         2       0.000000
         3       1.098612
         4       2.197225
```

### 1.4 for step five

from the above table from step four, we've calculated weights of each user, from that pick out those users, whose weight > 1.5

Output format:

| serial_number | user_name on mention | user_id | #tweets (no of tweets posted by user) | # retweets | weights > 1.5 |
|---|---|---|---|---|---|
| | | | | | |

```
In [15]: tw_data = tw_data[tw_data['log(retweets)'] > 1.5]
         tw_data.head()

Out[15]:          user        fullname     user_id  no_of_tweets retweets  \
         4   devbattles     Dev Battles  2377678050             2        9
         27   FollowMMA  Jason Chambers    22735770             1       14
         28     Doclach             Doc    21816418             1        6
         30    jedisct1     Frank Denis    17396038             2        6
         34    r_netsec       /r/netsec   238781296             1        7

             log(retweets)
         4         2.197225
         27        2.639057
         28        1.791759
         30        1.791759
         34        1.945910

In [16]: tw_data.reset_index(drop=True, inplace=True)
         tw_data.head()

Out[16]:          user        fullname     user_id  no_of_tweets retweets  \
         0  devbattles     Dev Battles  2377678050             2        9
         1   FollowMMA  Jason Chambers    22735770             1       14
         2     Doclach             Doc    21816418             1        6
         3    jedisct1     Frank Denis    17396038             2        6
         4    r_netsec       /r/netsec   238781296             1        7

            log(retweets)
         0       2.197225
         1       2.639057
         2       1.791759
         3       1.791759
         4       1.945910
```

### 1.5 for step six

In step five, count the number of users, # users are called as `Influential Users`

### 1.6 for step seven

For Influential users, calculate global influential score for each user.

$$Influentialscoreformula = \frac{noofretweets}{nooftweets}$$

```
In [17]: def inf_score(retweets, tweets):
             return (retweets / tweets)


         tw_data['inf_score'] = tw_data.apply(lambda x: inf_score(int(x.retweets), int(x.no_of_
         tw_data.head()
```

```
Out[17]:          user        fullname     user_id  no_of_tweets retweets  \
         0  devbattles     Dev Battles  2377678050             2        9
         1   FollowMMA  Jason Chambers    22735770             1       14
         2     Doclach             Doc    21816418             1        6
         3    jedisct1     Frank Denis    17396038             2        6
         4    r_netsec       /r/netsec   238781296             1        7

            log(retweets)  inf_score
         0       2.197225        4.5
         1       2.639057       14.0
         2       1.791759        6.0
         3       1.791759        3.0
         4       1.945910        7.0
```

### 1.7 for step eight

write down global influence scores in descending order and give rank to each influential user.

example: highest value of influential score = rank 1 . . . lowest value of influential score = rank n

Output format:

| global influential score in descending order | user name | global rank x_i |
|---|---|---|
| | | |

```
In [18]: tw_data = tw_data.sort_values(by=['inf_score'], ascending=False)

         tw_data.reset_index(drop=True, inplace=True)
         tw_data.head()
```

```
Out[18]:            user        fullname     user_id  no_of_tweets retweets  \
         0      benhamner      Ben Hamner    22674817             1       21
         1   randal_olson     Randy Olson    49413866             1       15
         2      FollowMMA  Jason Chambers    22735770             1       14
         3  jetrubyagency         JetRuby  3092433987             1       12
         4        arnicas     Lynn Cherny     6146692             1        9
```

```
   log(retweets)  inf_score
0       3.044522       21.0
1       2.708050       15.0
2       2.639057       14.0
3       2.484907       12.0
4       2.197225        9.0
```

### 1.8 for step nine

- collect tweets of the influential users from the output of step two
- count no of tweets posted by influential users

Output format:

| serial_number | screen_name | user_id | tweet_id | retweet_count | tweet |
|---|---|---|---|---|---|
| | | | | | |

```
In [28]: twitter_data_backup.loc[twitter_data_backup['user'] == "devbattles"]["Tweet"]

Out[28]: 84    #python game, join now: http://bit.ly/20gNQiră...
         93    #python game, join now: http://bit.ly/20gNQiră...
         Name: Tweet, dtype: object

In [34]: def get_tweet_from_user(username):
             return list(twitter_data_backup.loc[twitter_data_backup['user'] == username]["Twee

         get_tweet_from_user("devbattles")

Out[34]: ['#python game, join now: http://bit.ly/20gNQir\xa0pic.twitter.com/RGghBLzHc1',
          '#python game, join now: http://bit.ly/20gNQir\xa0pic.twitter.com/S88zDPoOgT']

In [40]: tw_data_backup = tw_data
         tw_data['Tweet'] = tw_data['user'].apply(lambda x: get_tweet_from_user(x))
         tw_data.head()

Out[40]:            user        fullname     user_id  no_of_tweets retweets  \
         0      benhamner      Ben Hamner    22674817             1       21
         1   randal_olson     Randy Olson    49413866             1       15
         2      FollowMMA  Jason Chambers    22735770             1       14
         3  jetrubyagency         JetRuby  3092433987             1       12
         4        arnicas     Lynn Cherny     6146692             1        9

            log(retweets)  inf_score                                          Tweet
         0       3.044522       21.0  [Want to learn machine learning with Python an...
         1       2.708050       15.0  [EasyAI: A #Python artificial intelligence fra...
         2       2.639057       14.0  [In an alternative universe #WarMachine hung h...
         3       2.484907       12.0  [#JS libraries to speed your development?\n11 ...
         4       2.197225        9.0  [Nice list of some good tricks in Pandas, for ...
```

### 1.9 for step Ten

- Convert tweets from textual format into numeric format by finding Tf - idf scores

Input : In step nine last column of the table (tweets of influential users)
Output format:

| tweet | word 1 | word 2 | ... | word n | abs(tvi) |
|---|---|---|---|---|---|
| tweet 1 | tv11 | tv12 | ... | tv1n | |
| tweet 2 | tv21 | tv22 | ... | tv2n | |
| ... | ... | ... | .. | ... | ... |
| ... | ... | ... | .. | ... | ... |
| ... | ... | ... | ... | ... | ... |
| ... | ... | ... | ... | ... | ... |

where tv11 = tf x idf
tf – term frequency
idf – inverse document frequecy
Term frequency Tf = number of times the word occurs in the tweet

$$InverseDocumentFrequency = \log(\frac{noofretweets}{nooftweets})$$