Twitter Analytics

June 4, 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.

[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:2007-07-06', 'python since:2007-07-06 until:2008-02-27',
'python since:2008-02-27 until:2008-10-20', 'python since:2008-10-20
until:2009-06-13', 'python since:2009-06-13 until:2010-02-04', 'python
since:2010-02-04 until:2010-09-29', 'python since:2010-09-29 until:2011-05-23',
'python since:2011-05-23 until:2012-01-14', 'python since:2012-01-14
until:2012-09-06', 'python since:2012-09-06 until:2013-04-30', 'python
since:2013-04-30 until:2013-12-22', 'python since:2013-12-22 until:2014-08-15',
'python since:2014-08-15 until:2015-04-09', 'python since:2015-04-09
until:2015-12-01', 'python since:2015-12-01 until:2016-07-24', 'python
since:2016-07-24 until:2017-03-17', 'python since:2017-03-17 until:2017-11-08',
'python since:2017-11-08 until:2018-07-02', 'python since:2018-07-02
until:2019-02-24']
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).
```

[29]:
```
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("/Users/zarwis/twitter_data/04June_0344.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()
```

[29]:
```
           fullname                                               html  \
0      James Dillon  <p class="TweetTextSize js-tweet-text tweet-te...
1   Triston Atilano  <p class="TweetTextSize js-tweet-text tweet-te...
2      Immortan Jay  <p class="TweetTextSize js-tweet-text tweet-te...
3             Alain  <p class="TweetTextSize js-tweet-text tweet-te...
4  ReadyList Sports  <p class="TweetTextSize js-tweet-text tweet-te...

                   id  likes  replies  retweets  \
0  1126275587415265280      1        1         0
1  1126275587398438912      0        1         0
2  1126275573129601024      1        0         0
3  1126275568847208448      4        2         2
4  1126275566313730049     23        0         7

                                                text            timestamp  \
0  Imagine thinking football isnt the greatest s...  2019-05-08T23:59:59
1  And the EPL season has been better than any of...  2019-05-08T23:59:59
2                          Already voted twice. Bink!  2019-05-08T23:59:56
3  The Lakers are the most dysfunctional team in ...  2019-05-08T23:59:55
4  ReadyList Sports is joining the @Broncos at th...  2019-05-08T23:59:54

                                        url             user
```

```
0      /misterdills/status/1126275587415265280      misterdills
1  /atilano_triston/status/1126275587398438912  atilano_triston
2  /1truemastermind/status/1126275573129601024  1truemastermind
3     /Alain_Patron/status/1126275568847208448     Alain_Patron
4  /ReadyListSports/status/1126275566313730049  ReadyListSports
```

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

[30]:
```
# 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
```

[30]:
```
Index(['fullname', 'id', 'retweets', 'text', 'timestamp', 'user'],
dtype='object')
```

[31]:
```
# renaming column names
# twitter_data.columns = ['Date', 'Tweet', 'user', 'retweets', 'fullname',␣
 ↪'Tweet_id']
twitter_data.columns = ['fullname', 'Tweet_id', 'retweets', 'Tweet', 'Date',␣
 ↪'user']

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.

[32]:
```
twitter_data = twitter_data[twitter_data.retweets != 0]
twitter_data.head()
```

[32]:
```
            fullname             Tweet_id  retweets  \
3              Alain  1126275568847208448         2
4   ReadyList Sports  1126275566313730049         7
12   Michael Brandon  1126275530796482561         9
13  OCNJSD Athletics  1126275529928249345         1
23             Alain  1126275568847208448         2
```

```
                                                 Tweet        Date  \
3   The Lakers are the most dysfunctional team in ...  2019-05-08
4   ReadyList Sports is joining the @Broncos at th...  2019-05-08
12  STARTUP 12 team Dynasty .5 PPR, 1QB, who at 1...  2019-05-08
13  Kasey Clifford, Coast Sports Today Player of t...  2019-05-08
23  The Lakers are the most dysfunctional team in ...  2019-05-08

                user
3       Alain_Patron
4    ReadyListSports
12           Mbb3303
13      OCRedRaiders
23      Alain_Patron
```

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)

[33]:
```
# 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()
```

[33]:
```
             fullname             Tweet_id  retweets             user
3               Alain  1126275568847208448         2     Alain_Patron
4   ReadyList Sports  1126275566313730049         7  ReadyListSports
12   Michael Brandon  1126275530796482561         9          Mbb3303
13  OCNJSD Athletics  1126275529928249345         1     OCRedRaiders
23              Alain  1126275568847208448         2     Alain_Patron
```

[34]:
```
# 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']
```

[35]:
```
def get_tweet_count(user: str) -> int:
    return tweet_count[user]
```

```
# get_tweet_count('mastercodeonlin')
```

[36]:
```
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="first")
twitter_data_without_tweet_count.reset_index(drop=True, inplace=True)
twitter_data_without_tweet_count.head()
```

[36]:

	fullname	Tweet_id	retweets	user	\
0	Alain	1126275568847208448	2	Alain_Patron	
1	ReadyList Sports	1126275566313730049	7	ReadyListSports	
2	Michael Brandon	1126275530796482561	9	Mbb3303	
3	OCNJSD Athletics	1126275529928249345	1	OCRedRaiders	
4	INEVITABLE	1126275388706050049	2	trapmoneybalvin	

	no_of_tweets
0	2
1	2
2	2
3	2
4	1

[38]:
```
# 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)
```

[39]:
```
# 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("Alain_Patron", api)
```

[39]:
```
169645026
```

[40]:
```
twitter_data_without_tweet_count['user_id'] =␣
 ↪twitter_data_without_tweet_count['user'].apply(lambda x:␣
 ↪int(get_user_id_from_screen_name(x, api)))
```

[41]:
```
twitter_data_without_tweet_count.head()
```

[41]:
```
           fullname             Tweet_id  retweets             user  \
0             Alain  1126275568847208448         2     Alain_Patron
1  ReadyList Sports  1126275566313730049         7  ReadyListSports
2   Michael Brandon  1126275530796482561         9          Mbb3303
3  OCNJSD Athletics  1126275529928249345         1     OCRedRaiders
4       INEVITABLE  1126275388706050049         2  trapmoneybalvin

   no_of_tweets              user_id
0             2            169645026
1             2  1006687252125134849
2             2           3121134618
3             2   877860881442504704
4             1  1014228766640484357
```

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)

[42]:
```
import math
twitter_data_without_tweet_count['log(retweets)'] =␣
 ↪twitter_data_without_tweet_count['retweets'].apply(lambda x: math.
 ↪log(int(x)))
twitter_data_without_tweet_count.head()
```

[42]:
```
           fullname             Tweet_id  retweets             user  \
0             Alain  1126275568847208448         2     Alain_Patron
1  ReadyList Sports  1126275566313730049         7  ReadyListSports
2   Michael Brandon  1126275530796482561         9          Mbb3303
3  OCNJSD Athletics  1126275529928249345         1     OCRedRaiders
```

```
4        INEVITABLE  1126275388706050049         2  trapmoneybalvin

   no_of_tweets              user_id  log(retweets)
0             2            169645026       0.693147
1             2  1006687252125134849       1.945910
2             2           3121134618       2.197225
3             2   877860881442504704       0.000000
4             1  1014228766640484357       0.693147
```

[43]:
```
tw_data = twitter_data_without_tweet_count[['user', 'fullname', 'user_id',␣
↪'no_of_tweets', 'retweets', 'log(retweets)']]
tw_data.head()
```

[43]:
```
              user          fullname              user_id  no_of_tweets  \
0     Alain_Patron             Alain            169645026             2
1  ReadyListSports  ReadyList Sports  1006687252125134849             2
2          Mbb3303   Michael Brandon           3121134618             2
3     OCRedRaiders  OCNJSD Athletics   877860881442504704             2
4  trapmoneybalvin        INEVITABLE  1014228766640484357             1

   retweets  log(retweets)
0         2       0.693147
1         7       1.945910
2         9       2.197225
3         1       0.000000
4         2       0.693147
```

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 @mention	user_id	#tweets (no of tweets posted by user)	# retweets	weights > 1.5

[44]:
```
tw_data = tw_data[tw_data['log(retweets)'] > 1.5]
tw_data.head()
```

[44]:
```
               user                                   fullname  \
1   ReadyListSports                           ReadyList Sports
2           Mbb3303                            Michael Brandon
8    hssportsawards  USA TODAY NETWORK High School Sports Awards
12   CanesOmbudsman                           budsy margarita,
15       ezralevant                             Ezra Levant ă
```

```
                 user_id  no_of_tweets  retweets  log(retweets)
1   1006687252125134849             2         7       1.945910
2            3121134618             2         9       2.197225
8     969631156940496896             1         5       1.609438
12           3241717645             1         5       1.609438
15             20878297             1        29       3.367296
```

[45]:
```
tw_data.reset_index(drop=True, inplace=True)
tw_data.head()
```

[45]:
```
              user                                     fullname  \
0  ReadyListSports                              ReadyList Sports
1          Mbb3303                               Michael Brandon
2   hssportsawards  USA TODAY NETWORK High School Sports Awards
3   CanesOmbudsman                              budsy margarita,
4       ezralevant                                Ezra Levant ã

                 user_id  no_of_tweets  retweets  log(retweets)
0  1006687252125134849             2         7       1.945910
1           3121134618             2         9       2.197225
2   969631156940496896             1         5       1.609438
3           3241717645             1         5       1.609438
4             20878297             1        29       3.367296
```

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.

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

[46]:
```
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_tweets)), axis=1)
tw_data.head()
```

[46]:
```
              user                                     fullname  \
0  ReadyListSports                              ReadyList Sports
1          Mbb3303                               Michael Brandon
2   hssportsawards  USA TODAY NETWORK High School Sports Awards
3   CanesOmbudsman                              budsy margarita,
4       ezralevant                                Ezra Levant ã
```

```
                user_id  no_of_tweets  retweets  log(retweets)  inf_score
0  1006687252125134849             2         7       1.945910        3.5
1           3121134618             2         9       2.197225        4.5
2   969631156940496896             1         5       1.609438        5.0
3           3241717645             1         5       1.609438        5.0
4             20878297             1        29       3.367296       29.0
```

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

[87]:
```
tw_data = tw_data.sort_values(by=['inf_score'], ascending=False)

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

[87]:
```
              user            fullname             user_id  no_of_tweets  \
0  stoolpresidente        Dave Portnoy            43775786             1
1          prageru             PragerU            41160276             1
2       clairlemon      Claire Lehmann          1398479138             1
3       comfiecore                remi  889216679368065024             1
4   BarSouthNCelly  Bar South N Celly           904108220             1

   retweets  log(retweets)  inf_score  \
0       630       6.445720      630.0
1       353       5.866468      353.0
2       161       5.081404      161.0
3       118       4.770685      118.0
4       104       4.644391      104.0

                                          tweet_data
0  {'1133160385752829953': 'This is what we wait ...
1  {'1130624137548845056': 'Men who identify as w...
2  {'1126997967678865408': 'Our dominant cultural...
3  {'1133160302894301186': 'you , small brain: ca...
4  {'1131348257626628096': 'Patrice Bergeron ahea...
```

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

[88]:
```
twitter_data_backup.loc[twitter_data_backup['user'] == "devbattles"]["Tweet_id"]
```

[88]:
```
Series([], Name: Tweet_id, dtype: object)
```

[89]:
```
def get_tweet_from_user(username) -> Dict:
    """ {"tweet_id": "Tweet"} """
    series_data = twitter_data_backup.loc[twitter_data_backup['user'] ==␣
 ↪username]
    return dict(zip(series_data["Tweet_id"], series_data["Tweet"]))
#     return list(twitter_data_backup.loc[twitter_data_backup['user'] ==␣
 ↪username]["Tweet"])

# get_tweet_from_user("devbattles")
```

[90]:
```
tw_data_backup = tw_data
tw_data['tweet_data'] = tw_data['user'].apply(lambda x: get_tweet_from_user(x))
tw_data.head()
```

[90]:
```
              user            fullname             user_id  no_of_tweets  \
0  stoolpresidente        Dave Portnoy            43775786             1
1          prageru             PragerU            41160276             1
2       clairlemon      Claire Lehmann          1398479138             1
3       comfiecore                remi  889216679368065024             1
4   BarSouthNCelly  Bar South N Celly           904108220             1

   retweets  log(retweets)  inf_score  \
0       630       6.445720      630.0
1       353       5.866468      353.0
2       161       5.081404      161.0
3       118       4.770685      118.0
4       104       4.644391      104.0

                                          tweet_data
0  {'1133160385752829953': 'This is what we wait ...
1  {'1130624137548845056': 'Men who identify as w...
2  {'1126997967678865408': 'Our dominant cultural...
3  {'1133160302894301186': 'you , small brain: ca...
4  {'1131348257626628096': 'Patrice Bergeron ahea...
```

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

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

[91]:
```
data_tweets = pd.DataFrame(tw_data["tweet_data"])

father_dict = dict()
for index, row in data_tweets.itertuples():
    father_dict.update(row)
# print(father_dict)
father_dict_keys = list(father_dict.keys())
father_dict_values = list(father_dict.values())

tweet_n_ids = pd.DataFrame.from_dict({'tweet_id':father_dict_keys, 'tweet':
 ↪father_dict_values})

print(len(tweet_n_ids))
tweet_n_ids.head()
```

```
291
```

[91]:

	tweet_id	tweet
0	1133160385752829953	This is what we wait all season for. Its wha...
1	1130624137548845056	Men who identify as women are now dominating w...
2	1126997967678865408	Our dominant cultural narratives are based on ...
3	1133160302894301186	you , small brain: catra is a delinquent and a...
4	1131348257626628096	Patrice Bergeron ahead of the #StanleyCupFinal...

[92]:
```
import re

def clean_tweet(tweet: str) -> List:
    """
    1. Remove RETWEET Tags 'RT @'
```

```
    2. Remove '@mention' tags
    3. Split the Tweet, change the case to lower()
    4. Remove single and double quotes from the words
    5. ignore words with 'http' or 'https://'
    6. Remove digits <numbers> from words
    7. Ignore words with pic.twitter.com
    8. Remove special characters
    """
    words_list = []
    # remove retweet tags 'RT @' & '@mention' tags & '#tag'
    if "RT @" in tweet:
        tweet = re.sub("RT @[A-Za-z0-9:]+\s", "", tweet)

    if "@" in tweet:
        tweet = re.sub("@[A-Za-z0-9]+\s", "", tweet)

    if "#" in tweet:
        tweet = re.sub("#[A-Za-z0-9]+\s", "", tweet)

    tweet = tweet.strip("")
    tweet = tweet.replace("", "")
    tweet = tweet.replace("'", "")
    tweet = tweet.replace("~", "")

    # Split the tweet, change the case to lower()
    tweet = tweet.lower()
    words = tweet.split()

    for word in words:
        # ignore words with 'http' or 'https://' or 'pic.twitter.com'
        if not (word.startswith("http") or word.startswith("https://")\
                or word.startswith("pic.twitter.com/") or "https" in word \
                or "twitter.com" in word):
            # Remove special characters
            word = "".join([i for i in word if ord(i) in range(97, 123)])
            if word:
                words_list.append(word)

    return words_list

clean_tweet("Grab your Yoga mat take a cruise on the harbour and let the kids␣
 ↪run bare feet at school today - our tribute to @NSWRL series winning coach␣
 ↪Brad Fittler @telegraph_sport https://www.dailytelegraph.com.au/sport/nrl/
 ↪state-of-origin/
 ↪brad-fittlers-brave-overhaul-of-the-blues-has-paid-off-in-the-sweetest-way-possible/
 ↪news-story/73f39bd3de7d5caa89c76080cae6ebf9ã")
```

[92]:
```
['grab',
 'your',
 'yoga',
 'mat',
 'take',
 'a',
 'cruise',
 'on',
 'the',
 'harbour',
 'and',
 'let',
 'the',
 'kids',
 'run',
 'bare',
 'feet',
 'at',
 'school',
 'today',
 'our',
 'tribute',
 'to',
 'series',
 'winning',
 'coach',
 'brad',
 'fittler',
 'telegraphsport']
```

[93]:
```python
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer


def remove_stop_words(words_list: List) -> List:
    stop_words = stopwords.words('english')
    # print(len(words_list))
    words_list = [i for i in words_list if i not in stop_words]
    # print(len(words_list))
    return words_list


def stem_tweets(words_list: List) -> List:
    ps = PorterStemmer()
    words_list = [ps.stem(i) for i in words_list]
    words_list = list(set(words_list))
```

```
    return words_list


def lemmatize_tweets(words_list: List) -> List:
    lm = WordNetLemmatizer()
    words_list = [lm.lemmatize(i) for i in words_list]
    words_list = list(set(words_list))
    return words_list


lemmatize_tweets(stem_tweets(remove_stop_words(clean_tweet("Grab your Yoga mat␣
 ↪take a cruise on the harbour and let the kids run bare feet at school today␣
 ↪- our tribute to @NSWRL series winning coach Brad Fittler @telegraph_sport␣
 ↪https://www.dailytelegraph.com.au/sport/nrl/state-of-origin/
 ↪brad-fittlers-brave-overhaul-of-the-blues-has-paid-off-in-the-sweetest-way-possible/
 ↪news-story/73f39bd3de7d5caa89c76080cae6ebf9 "))))
```

[93]:
```
['grab',
 'brad',
 'yoga',
 'tribut',
 'cruis',
 'seri',
 'today',
 'take',
 'coach',
 'school',
 'harbour',
 'foot',
 'fittler',
 'win',
 'kid',
 'telegraphsport',
 'bare',
 'let',
 'mat',
 'run']
```

[94]:
```
def get_tweet_words(tweet: str) -> List:
    """
    1. clean the tweet
    2. remove stop words
    3. stem the words
    4. lemmatize the words
    """
    words = clean_tweet(tweet)
    words = remove_stop_words(words)
    words = stem_tweets(words)
```

```
    words = lemmatize_tweets(words)

    return words

tweet_n_ids['tweet_words'] = tweet_n_ids['tweet'].apply(lambda x:␣
 ↪get_tweet_words(x))
tweet_n_ids.head()
```

[94]:
```
              tweet_id                                              tweet  \
0  1133160385752829953  This is what we wait all season for.  Its wha...
1  1130624137548845056  Men who identify as women are now dominating w...
2  1126997967678865408  Our dominant cultural narratives are based on ...
3  1133160302894301186  you , small brain: catra is a delinquent and a...
4  1131348257626628096  Patrice Bergeron ahead of the #StanleyCupFinal...

                                        tweet_words
0  [gruel, within, new, season, post, enjoy, grow...
1  [identifi, woman, movement, feminist, transgen...
2  [also, narr, biolog, sure, appli, domin, syste...
3  [delinqu, scholarship, she, grade, small, stra...
4  [bergeron, patric, kid, skate, tsnsport, ahead...
```

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

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

[95]:
```
def inverse_doc_freq(retweets: int, tweets: int) -> float:
    """ returns a floating point number
        Inverse Document Frequency = log(retweets / tweets)
    """
    return math.log(retweets/tweets)


def term_freq(word, tweet_id):
    words = tweet_n_ids.loc[tweet_n_ids['tweet_id'] == tweet_id]["tweet_words"]
    words = list(words)[0]
    if word in words:
        return words.count(word)
    else:
        return 0

# inverse_doc_freq(10, 5)
term_freq("kid", "1131348257626628096")
```

[95]:
```
1
```

- make tweet dump

- calculate inverse document frequency
- calculate term frequency

[97]:
```
tweet_dump = []
for row in tweet_n_ids.itertuples():
    tweet_dump += row.tweet_words
print(len(tweet_dump))
# print(tweet_dump)
```

```
4185
```

[98]:
```
for row in tweet_n_ids.itertuples():
    tweet_id = row.tweet_id
#     retweets = row.retweets
#     tweets = row.tweets
    for word in tweet_dump:
        pass
```