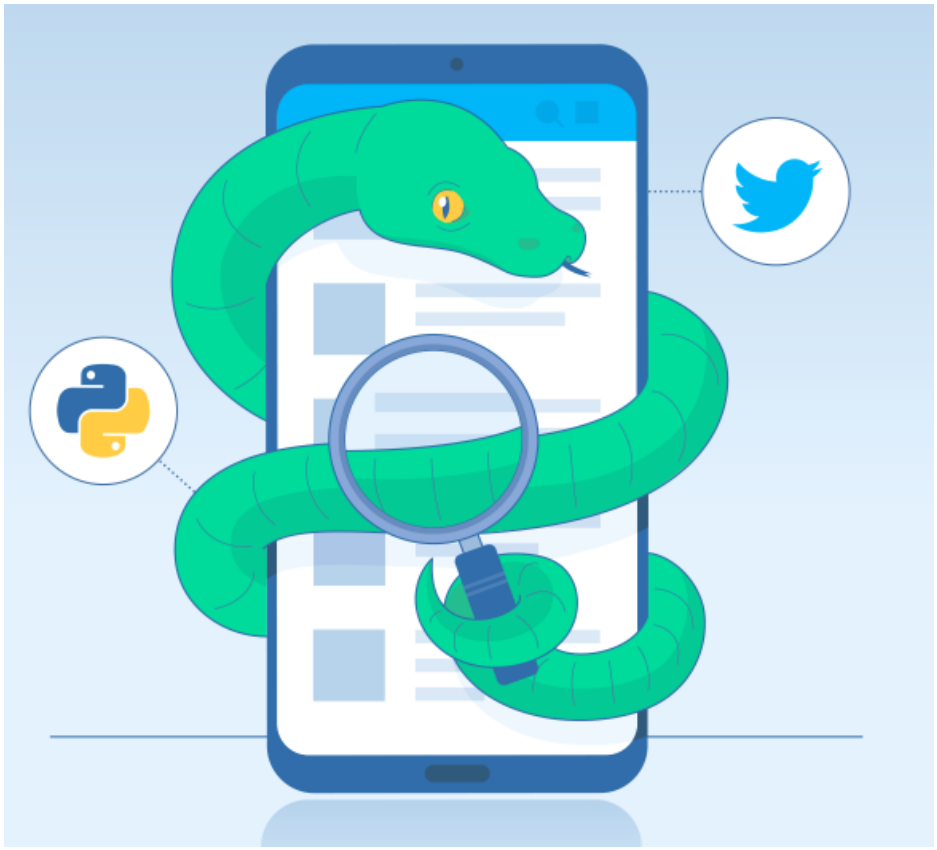
Extracting Twitter Data, Pre-Processing and Sentiment Analysis using Python 3.0

Dilan Jayasekara Follow
Apr 3 · 8 min read ★



Twitter Data Extraction using Python

Twitter is a gold mine of data. Unlike other social platforms, almost every user's tweets are completely public and pullable. This is a huge plus if you're trying to get a large amount of data to run analytics on. Twitter data is also pretty specific. Twitter's API allows you to do complex queries like pulling every tweet about a certain topic within the last twenty minutes or pull a certain user's non-retweeted tweets. [Source: https://chatbotslife.com/twitter-data-mining-a-guide-to-big-data-analytics-using-python-4efc8ccfa219]

Hereby in this article, I'll guide you through the steps I did to extract three set of Twitter data uniquely separated by three set of keywords + hashtags.

> *If you come up to find any issues in the code feel free to ask*

1. **Import Libraries**

```
from tweepy import Stream
from tweepy import OAuthHandler
from tweepy.streaming import StreamListener
import json
import pandas as pd
import csv
import re #regular expression
```

```
from textblob import TextBlob
import string
import preprocessor as p
```

2. Twitter credentials

If you have no Idea what these twitter credentials are, you must become a twitter developer to use these and I'm sure you'll find plenty of tutorials here in Medium and also youtube got heaps of videos of how to do this.

```
#Twitter credentials for the app
consumer_key = 'xxxxx'
consumer_secret = 'xxxx'
access_key= 'xxxx'
access_secret = 'xxxx'
```

3. Credentials

Pass these credentials to Tweepy's OAuthHandler instance named '**auth**', then using that instance call the method **set_access_token** by passing the above-created **access_key** and, **access_secret.**

```
#pass twitter credentials to tweepy


auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_key, access_secret)
api = tweepy.API(auth)
```

4. What we extract from Twitter and Why?

Ok, First things are done. I'll quickly disclose what I'm attempting to extricate from twitter and will reveal to you a little anecdote about that. I'm trying to connect the relation between two issues and a practical solution to both of those issues by using Twitter data.

Two major diseases were taken as my two issues: Heart Stroke Twitter data & Epilepsy twitter data

Solution: Telemedicine

Simply keep the picture in your mind and in the long run you'll understand what I'm trying to explain here.

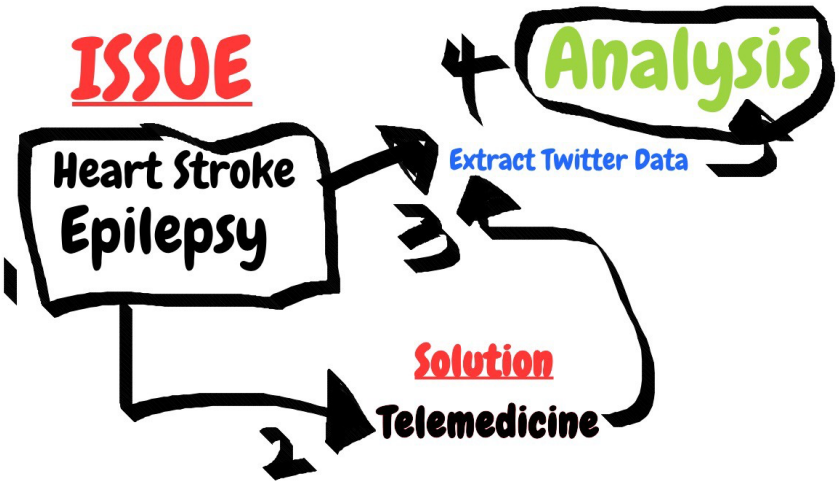


Figure 1.0: Basic Idea behind the analysis

5. Create file paths for the 3 CSV files

```
#declare file paths as follows for three files
```

```
telemedicine_tweets =
"data/telemedicine_data_extraction/telemedicine_data.csv"
```

```
epilepsy_tweets =
"data/telemedicine_data_extraction/epilepsy_data.csv"


heart_stroke_tweets =
"data/telemedicine_data_extraction/heart_stroke_tweets_data
.csv"
```

6. What exactly we need to extract?

Columns, Yes I meant that. Which columns do we **need the most** for our analysis?

No	Data Field	Description
1	**id**	Unique id of the tweet
2	**created_at**	Tweet date and Time
3	**source**	Source of the tweet (Via web/Android/iPhone)
4	**text**	Tweet Text
5	**sentiment**	Sentiment of the tweet
	polarity	Separated the polarity from Sentiment
	subjectivity	Separated the subjectivity from Sentiment
6	**lang**	Language used in the tweet
7	**favorite_count**	Number of favorites per tweet
8	**retweet_count**	Number of retweets
9	**original_author**	Profile user name of the tweet's author
10	**possibly_sensitive**	Sensitivity of the message (Boolean true / false)
11	**Hashtags**	Extracted all the hashtags in the tweet
12	**User_mentions**	Any other profile mentions in the tweets
13	**Place**	User's location
14	**Place_coord_boundaries**	Coordinates of the tweet's location (if applicable)

Figure 2: Data Fields

```
#columns of the csv file
```

```
COLS = ['id', 'created_at', 'source',
'original_text','clean_text',
'sentiment','polarity','subjectivity', 'lang',
'favorite_count', 'retweet_count', 'original_author',
'possibly_sensitive', 'hashtags',
'user_mentions', 'place', 'place_coord_boundaries']
```

7. Handle Emoticons and Emojis

7.1 Emoticons: Let's declare a series of emoticons (Happy & Sad) because we don't need the old school emoticons in the middle of a

sentence blocking us against our sentiment analysis.

```
#HappyEmoticons
emoticons_happy = set([
    ':-)', ':)', ';)', ':o)', ':]', ':3', ':c)', ':>',
'=]', '8)', '=)', ':}',
    ':^)', ':-D', ':D', '8-D', '8D', 'x-D', 'xD', 'X-D',
'XD', '=-D', '=D',
    '=-3', '=3', ':-))', ":'-)", ":')", ':*', ':^*', '>:P',
':-P', ':P', 'X-P',
    'x-p', 'xp', 'XP', ':-p', ':p', '=p', ':-b', ':b',
'>:)', '>;)', '>:-)',
    '<3'
    ])
```

. . .

```
# Sad Emoticons
emoticons_sad = set([
    ':L', ':-/', '>:/', ':S', '>:[', ':@', ':-(', ':[', ':-
||', '=L', ':<',
    ':-[', ':-<', '=\\', '=/', '>:(', ':(', '>.<', ":'-(",
":'(", ':\\', ':-c',
    ':c', ':{', '>:\\', ';('
    ])
```

. . .

7.2 Emoji Recognition

Because that's a must, nowadays people don't tweet without emojis, as in a matter of fact it became another language, especially between teenagers so have to come up with a plan to do so.

```
#Emoji patterns
emoji_pattern = re.compile("["
         u"\U0001F600-\U0001F64F"  # emoticons
         u"\U0001F300-\U0001F5FF"  # symbols & pictographs
         u"\U0001F680-\U0001F6FF"  # transport & map
symbols
```

```
            u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
            u"\U00002702-\U000027B0"
            u"\U000024C2-\U0001F251"
            "]+", flags=re.UNICODE)
```

And then we combine both happy and sad emoticon array-lists first:

```
#combine sad and happy emoticons

emoticons = emoticons_happy.union(emoticons_sad)
```

8. Method to Clean (Preprocessor)

Preprocessing here is done by two methods:

Method1: Using tweet-preprocessor Preprocessor is a preprocessing library for tweet data written in Python. When building Machine Learning systems based on tweet data, a preprocessing is required. This library makes it easy to clean, parse or tokenize the tweets.

Method2: I've manually defined a function to double check and our tweet preprocessing and it's always better to be sure that our data is cleaned 100%.

8.1 Method-1

```
import preprocessor as p
```

PS: I have already imported this in Step 1 (Import Libraries section)

Example:

```
clean_text = p.clean(twitter_text)
```

8.2 Method-2

Declare a method called clean_tweets(tweet) and this method will clean some remains of the twitter data which is left undone by tweet-preprocessor and double check emoticons and emoji's because some older version of mobile's emoticons is not supported in tweet preprocessor's clean method (Method1).

```
def clean_tweets(tweet):

    stop_words = set(stopwords.words('english'))
    word_tokens = word_tokenize(tweet)

#after tweepy preprocessing the colon symbol left remain
after      #removing mentions
    tweet = re.sub(r':', '', tweet)
    tweet = re.sub(r',Ä¶', '', tweet)
#replace consecutive non-ASCII characters with a space
    tweet = re.sub(r'[^\x00-\x7F]+',' ', tweet)

#remove emojis from tweet
    tweet = emoji_pattern.sub(r'', tweet)


#filter using NLTK library append it to a string
    filtered_tweet = [w for w in word_tokens if not w in
stop_words]
    filtered_tweet = []

#looping through conditions
    for w in word_tokens:
#check tokens against stop words , emoticons and
punctuations
        if w not in stop_words and w not in emoticons and w
not in string.punctuation:
            filtered_tweet.append(w)
    return ' '.join(filtered_tweet)
    #print(word_tokens)
    #print(filtered_sentence)return tweet
```

> *At this point, I want you to give your attention on Stop Words, and why is it important for Text Mining. And for now, Don't look at the code inside our method, you can see and try to understand this at the end of* ***Step-9***

9. Extract Tweets

To connect to Twitter's API, we will be using a Python library called Tweepy, which is an excellently supported tool for accessing the Twitter API. It supports Python 2.6, 2.7, 3.3, 3.4, 3.5, and 3.6. There are some other Twitter API's also but I recommend Tweepy since it never gave any trouble.

I'll post the full code below and section each important part and describe what that part is for:

9.1 Beginning of the method

```
def write_tweets(keyword, file):
    #If the file exists, then read the existing data from
the CSV file.
    if os.path.exists(file):
        df = pd.read_csv(file, header=0)
    else:
        df = pd.DataFrame(columns=COLS)
    #page attribute in tweepy.cursor and iteration
    for page in tweepy.Cursor(api.search, q=keyword,
                              count=200, include_rts=False,
```

In this method, I've created **two parameters**; one for the file name hence we have three different files to take care of as well as three sets of keywords for those file to be filled with which explains the second parameter.

9.2 JSON

The result you receive from the Twitter API is in a JSON format and has quite an amount of information attached.

```
for status in page:
            new_entry = []
            status = status._json
```

```
if status['lang'] != 'en':
                continue
```

We create this array of string name new_entry=[] to store all the JSON parsed data on each iteration. and we continue to retrieve data only and if the language is English since I don't need any trouble translating tweet language to language at this stage. :)

9.3 Replace RT's and FAVs

*when running the code, **below code replaces the retweet amount and number of favorites that are changed since the last download.**

```
if status['created_at'] in df['created_at'].values:
  i = df.loc[df['created_at'] ==
status['created_at']].index[0]
  if status['favorite_count'] != df.at[i, 'favorite_count']
or \
  status['retweet_count'] != df.at[i, 'retweet_count']:
  df.at[i, 'favorite_count'] = status['favorite_count']
  df.at[i, 'retweet_count'] = status['retweet_count']
 continue
```

9.4 Time for preprocessing

- Now's the time for us to use the **Method-1** of tweet preprocessing.

```
clean_text = p.clean(status['text'])
```

- Call clean_tweet method-2 for extra preprocessing

```
filtered_tweet=clean_tweets(clean_text)
```

9.5 Sentiment

The `sentiment` property returns a named tuple of the form `Sentiment(polarity,subjectivity)`. The polarity score is a float within

the range [-1.0, 1.0]. The subjectivity is a float within the range [0.0, 1.0] where 0.0 is very objective and 1.0 is very subjective.

- Pass the filtered_tweet to TextBlob for sentiment calculation and I separately stored **sentiment**, **polarity**, and **subjectivity** in **three** different variables.

```
blob = TextBlob(filtered_tweet)
```

```
Sentiment = blob.sentiment
            polarity = Sentiment.polarity
            subjectivity = Sentiment.subjectivity
```

9.6 Append ALL

Append the JSON parsed data to the string array we created:

```
new_entry += [status['id'], status['created_at'],
              status['source'],
status['text'],filtered_tweet,
Sentiment,polarity,subjectivity, status['lang'],
              status['favorite_count'],
status['retweet_count']]
```

Those appended data is the data we've already extracted from twitter Using Tweepy. But there are much more data fields as I mentioned in **Step-6.** In order to follow the sequence we gather the original author's user name (Twitter profile name of the tweet)

```
new_entry.append(status['user']['screen_name'])
```

9.7 Tweets With NSFW Content

possibly_sensitive column of the tweet data is for NSFW content on Twitter.

```
try:
                is_sensitive = status['possibly_sensitive']
            except KeyError:
                is_sensitive = None
            new_entry.append(is_sensitive)
```

9.8 Hashtags and other user mentions of the tweet

```
hashtags = ", ".join([hashtag_item['text'] for hashtag_item
in status['entities']['hashtags']])

new_entry.append(hashtags) #append the hashtags

mentions = ", ".join([mention['screen_name'] for mention in
status['entities']['user_mentions']])

new_entry.append(mentions) #append the user mentions
```

9.9 Get the location of the tweet

I'm trying to track down the location of the tweet but practically this is a bit hard and nearly impossible because **most of the users don't allow** Twitter to access their location always. But anyway I went with something like this:

```
try:
                coordinates = [coord for loc in
status['place']   ['bounding_box']['coordinates'] for coord
in loc]
            except TypeError:
                coordinates = None
            new_entry.append(coordinates)
```

Hence the difficulty that we have to face while extracting the 'tweet-location', I've managed to get the **user's profile location instead of the tweet's location** since it basically gives me an idea of the region and country the user is located.

```
try:
                location = status['user']['location']
            except TypeError:
                location = ''
            new_entry.append(location)
```

9.10 Finish data gathering

Almost done. We now have all the data we need, let's nicely wrap it up to a data frame.

```
single_tweet_df = pd.DataFrame([new_entry], columns=COLS)
df_final = df.append(single_tweet_df, ignore_index=True)
```

10. Write into CSV file

```
csvFile = open(file, 'a' ,encoding='utf-8')

df.to_csv(csvFile, mode='a', columns=COLS, index=False,
encoding="utf-8")
```

to_csv is used to write all the data we gathered into the particular CSV file and make sure to include ***encoding="utf-8"*** otherwise some problems might occur while running operations on data in the CSV file. If so, we have to encode it manually so it's better to do the right thing in the first place.

11. Declare Keywords

```
telemedicine_keywords = '#telemedicine OR #telehealth OR
#digitalhealth OR #ehealth OR #digitalpatient OR
#digitaltransformation'


Epilepsy_keywords = '#Epilepsy OR #epilepsyawareness OR
#epilepsyaction OR #epilepsyalerts OR #epilepsybed OR
#epilepsycongres OR #epilepsysurgery OR #epilepsysurgery OR
#Epilepsytreatment OR #seizures OR #seizurefree'

HeartDisease_keywords = '#HeartDisease OR #stroke OR
#Stroking OR #strokepatient OR #StrokeSurvivor OR
#hearthealth OR #Stroke OR #HeartFailure'
```

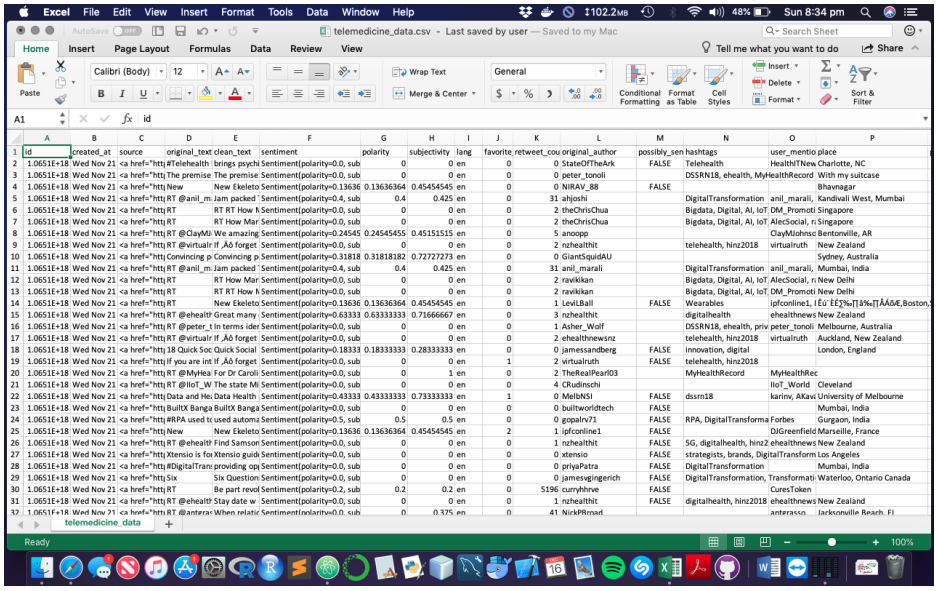
12. Call our method

```
write_tweets(telemedicine_keywords, telemedicine_tweets)
write_tweets(Epilepsy_keywords, epilepsy_tweets)
write_tweets(HeartDisease_keywords, heart_stroke_tweets)
```

> *Click here to access the full source code.*

13. Sneak peek into the CSV Files we created:

Created CSV files



Preview of a CSV file

In my next article, I'll share how I analyzed those data and how can we visualize data using Python library Matplotlib.

Skol!

Twitter: https://twitter.com/dylankalpaLinkedIn: https://www.linkedin.com/in/dilankalpa/