SpotleCovidAnalysis

June 6, 2021

1 Spotle Covid Twitter Data Analysis

1.1 - Rohit Narayanan

1.1.1 - National Institute of Technology Puducherry

2 Introduction

This is a complete extravagant report of the Twitter Data related to covid times. The dataset is analyzed using Python and constructive conclusions and inferences are reached at. This report presents both the **analytical side and the inferential side**, thus being an inseparable combo in the same file.

3 1) Installing PyDrive for uploading raw data

We import GoogleAuth, GoogleDrive, auth, GoogleCredentials modules for easy data upload from our Google drives. Then we authenticate and create a PyDrive client. We then click on the link, authenticate with our Gmail account and copy the generated code into the given space, to get ourselves authenticated. Now get the GDrive link for our file, and get the contents in to our workspace.

[24]:
```
!pip install -U -q PyDrive

from pydrive.auth import GoogleAuth

from pydrive.drive import GoogleDrive

from google.colab import auth

from oauth2client.client import GoogleCredentials
```

[2]:
```
auth.authenticate_user()

gauth = GoogleAuth()

gauth.credentials = GoogleCredentials.get_application_default()

drive = GoogleDrive(gauth)
```

[3]:
```
link='https://drive.google.com/file/d/1xy8BBnINILiZMFSwpzC8Rq3KwUEHBLjo/view?
 ↪usp=sharinghttps://drive.google.com/file/d/1xy8BBnINILiZMFSwpzC8Rq3KwUEHBLjo/
 ↪view?usp=sharing'

import pandas as pd

id = link.split("/")[-2]

downloaded = drive.CreateFile({'id':id})

downloaded.GetContentFile('CovidTwitterAnalysis.csv')

df = pd.read_csv('CovidTwitterAnalysis.csv')

print(df)
```

```
                           created_at  ... user_statuses_count
0      Wed Mar 25 06:20:02 +0000 2020  ...                1770
1      Wed Mar 25 06:36:25 +0000 2020  ...                  73
2      Wed Mar 25 06:18:32 +0000 2020  ...                6002
3      Wed Mar 25 06:05:46 +0000 2020  ...                1387
4      Wed Mar 25 06:31:10 +0000 2020  ...                  13
...                               ...  ...                 ...
44174  Thu May 07 03:46:30 +0000 2020  ...                9457
44175  Thu May 07 02:57:32 +0000 2020  ...                3433
44176  Thu May 07 03:47:42 +0000 2020  ...               23734
44177  Thu May 07 03:48:04 +0000 2020  ...               15185
44178  Thu May 07 03:48:08 +0000 2020  ...                6931

[44179 rows x 19 columns]
```

4 2) Importing required Python libraries and modules

Tweepy - Python library for accessing the Twitter API.
TextBlob - Python library for processing textual data
WordCloud - Python library for creating image wordclouds
Pandas - Data manipulation and analysis library
NumPy - mathematical functions on multi-dimensional arrays and matrices
Regular Expression Python module
Matplotlib - plotting library to create graphs and charts
Settings for Matplotlib graphs and charts
nltk - for text manipulation

[4]:
```
import tweepy

from textblob import TextBlob
from wordcloud import WordCloud

import pandas as pd

import numpy as np

import re

from ipywidgets import *

import seaborn as sns

import matplotlib.pyplot as plt

from pylab import rcParams

rcParams['figure.figsize'] = 12, 8

import string

import nltk

import warnings

warnings.filterwarnings("ignore", category=DeprecationWarning)

import mpl_toolkits

import io

%matplotlib inline
```

5 3) Characteristics of the data. How the given data looks

head()-This function returns the first n rows for the object based on position. It is useful for quickly testing if your object has the right type of data in it.

describe()- It is used to generate descriptive statistics that summarize the central tendency, dispersion and shape of a dataset's distribution.

dtypes-It returns a Series with the data type of each column.

shape()- This gets the number of rows and columns

drop()-It removes rows or columns by specifying label names and corresponding axis, or by specifying directly index or column names

duplicated()- It returns a boolean Series denoting duplicate rows.

count()- It counts non-NA cells for each column or row for any specified condition.

drop_duplicates() -It return DataFrame with duplicate rows removed.

[5]:
```
df.head()
```

[5]:
```
                        created_at  ... user_statuses_count
0  Wed Mar 25 06:20:02 +0000 2020  ...                1770
1  Wed Mar 25 06:36:25 +0000 2020  ...                  73
2  Wed Mar 25 06:18:32 +0000 2020  ...                6002
3  Wed Mar 25 06:05:46 +0000 2020  ...                1387
4  Wed Mar 25 06:31:10 +0000 2020  ...                  13

[5 rows x 19 columns]
```

[6]:
```
df.describe()
```

[6]:
```
       favorite_count            id  ...  user_listed_count  user_statuses_count
count    44179.000000  4.417900e+04  ...       44179.000000         4.417900e+04
mean         8.688834  1.249746e+18  ...          15.648951         1.830381e+04
std        690.196275  5.513385e+15  ...          86.748546         5.619829e+04
min          0.000000  1.242693e+18  ...           0.000000         1.000000e+00
25%          0.000000  1.244130e+18  ...           0.000000         3.990000e+02
50%          0.000000  1.250482e+18  ...           0.000000         2.476000e+03
75%          1.000000  1.256266e+18  ...           5.000000         1.330600e+04
max     144012.000000  1.258242e+18  ...        5775.000000         2.117851e+06

[8 rows x 8 columns]
```

[7]:
```
df.dtypes
```

[7]:
```
created_at                object
hashtags                  object
favorite_count             int64
id                         int64
lang                      object
place                     object
retweet_count              int64
text                      object
tweet_url                 object
user_screen_name          object
user_description          object
user_favourites_count      int64
user_followers_count       int64
user_friends_count         int64
user_listed_count          int64
user_location             object
user_name                 object
user_screen_name.1        object
user_statuses_count        int64
dtype: object
```

[8]:
```
df = df.drop(['created_at', 'id', 'lang', 'place', 'tweet_url',␣
 ↪'user_screen_name','user_description' ,'user_location', 'user_name',␣
 ↪'user_screen_name.1'], axis=1)
```

[9]:
```
df.shape
```

[9]:
```
(44179, 9)
```

[10]:
```
duplicate_rows_df = df[df.duplicated()]

print("number of duplicate rows: ", duplicate_rows_df.shape)
```

```
number of duplicate rows:  (105, 9)
```

[11]:
```
df.count()
```

[11]:
```
hashtags                 17907
favorite_count           44179
retweet_count            44179
text                     44179
user_favourites_count    44179
user_followers_count     44179
user_friends_count       44179
user_listed_count        44179
user_statuses_count      44179
dtype: int64
```

[12]:
```
df = df.drop_duplicates()
```

[13]:
```
df.count()
```

[13]:
```
hashtags                 17873
favorite_count           44074
retweet_count            44074
text                     44074
user_favourites_count    44074
user_followers_count     44074
user_friends_count       44074
user_listed_count        44074
user_statuses_count      44074
dtype: int64
```

6 4) Detecting and analyzing Subjectivity and Polarity of the tweets

This creates a function that determines subjectivity and polarity from the textblob package and then apply these functions to the dataframe. Then we build a function to calculate and categorize each tweet as Positive, Neutral, and Negative. Finally we create another column "Score" and apply the function to the dataframe.

Then we move on to visualizing and summarizing the data.

Here we will be having the following

1) Bar plot - Polarity

2) Scatter plot - Subjectivity vs Polarity

3) Number of subjective and objective tweets

4) Number of positive, negative and neutral tweets.

[14]:
```
def getTextSubjectivity(txt):
    return TextBlob(txt).sentiment.subjectivity

def getTextPolarity(txt):
    return TextBlob(txt).sentiment.polarity
```

[15]:
```
df['Subjectivity'] = df['text'].apply(getTextSubjectivity)

df['Polarity'] = df['text'].apply(getTextPolarity)
```

[16]:
```
df = df.drop(df[df['text'] == ''].index)

df.head(50)
```

[16]:
```
                                               hashtags  ...  Polarity
0                                                   NaN  ...  0.125000
1                                                   NaN  ...  0.000000
2                                                   NaN  ...  0.000000
3                            SSC_UFM_MAT_KARONA Corona UFM  ...  0.033333
4                          Corona pritamkumarmurari Voice  ...  0.600000
5                                                   NaN  ...  0.000000
6                                                   NaN  ... -0.125000
7                                                   NaN  ...  0.080952
8                                                   NaN  ...  0.600000
9                                                   NaN  ...  0.333333
10  India IndiaFightsCorona Delhi DelhiFightsCoron...  ...  0.200000
11                                                  NaN  ...  0.127483
12  cabinetmeeting Social_Distancing StayHome stay...  ...  0.225000
13  COVIDIOTS corona Twitter KCR CoronavirusPandem...  ... -0.143750
14                                                  NaN  ... -0.125000
15                                                  NaN  ...  0.083333
16       homemadefoods corona healthyfood selfcooking  ...  0.000000
17                                   KeralaFightsCorona  ...  0.000000
18                                                  NaN  ...  0.800000
19                                                  NaN  ...  0.111111
20                                                  NaN  ...  0.284091
21                                                  NaN  ... -0.400000
22              corona coronavirus quarantine lockdown  ... -0.100000
23  lockdown Delhi corona delhipolice stayathome e...  ...  0.350000
24                                                  NaN  ... -0.050000
25  Day3ofQuarantine QuarantineLife 21daylockdown ...  ...  0.000000
26                                                  NaN  ... -0.100000
27                 harharmahadev jaimahakala shivshambho  ...  1.000000
28                                                  NaN  ... -0.200000
29                                                  NaN  ...  0.000000
```

```
30                                                  NaN  ...  0.000000
31                                               Corona  ... -0.060714
32                                                  NaN  ...  0.125926
33  CoronavirusOutbreakindia CoronavirusPandemic C...  ...  0.250000
34                                                  NaN  ...  0.000000
35                                                  NaN  ...  0.127483
36          Corona StayAtHome CoronaVirus RedFMTelugu  ...  0.000000
37  nature slowdown corona QuaratineLife Quarantin...  ...  0.000000
38                                               Corona  ...  0.500000
39                                                  NaN  ...  0.000000
40                                        IndiaStayHome  ...  0.650000
41                                                  NaN  ...  0.333333
42                           Corona SocialDistancing  ...  0.100000
43                                                  NaN  ...  0.102500
44                                                  NaN  ...  0.000000
45                                     IndiaFightsCoron  ...  0.075000
46  coronavrus corona mat corona airqualityindex...  ...  0.357143
47                                                  NaN  ...  0.625000
48  stayathome eathealthy corona healthfood homema...  ...  0.600000
49                                                  NaN  ... -0.025000

[50 rows x 11 columns]
```

[17]:
```
def getTextAnalysis(a):
    if a < 0:
        return "Negative"
    elif a == 0:
        return "Neutral"
    else:
        return "Positive"
```

[18]:
```
df['Score'] = df['Polarity'].apply(getTextAnalysis)
```

[19]:
```
positive = df[df['Score'] == 'Positive']

print(str(positive.shape[0]/(df.shape[0])*100) + " % of positive tweets")
```

```
47.256886146027135 % of positive tweets
```

[20]:
```
negative = df[df['Score'] == 'Negative']

print(str(negative.shape[0]/(df.shape[0])*100) + " % of negative tweets")
```

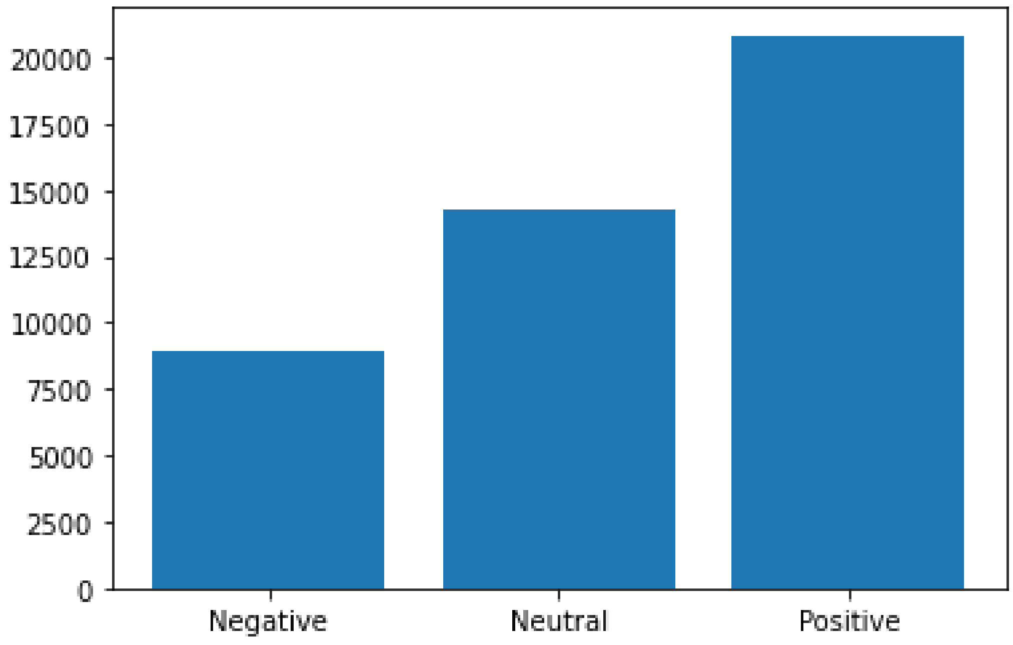
```
20.28406770431547 % of negative tweets
```

[21]:
```
neutral = df[df['Score'] == 'Neutral']

print(str(neutral.shape[0]/(df.shape[0])*100) + " % of neutral tweets")
```

```
32.4590461496574 % of neutral tweets
```

[22]:
```
labels = df.groupby('Score').count().index.values

values = df.groupby('Score').size().values

plt.bar(labels, values)
```

[22]:
```
<BarContainer object of 3 artists>
```

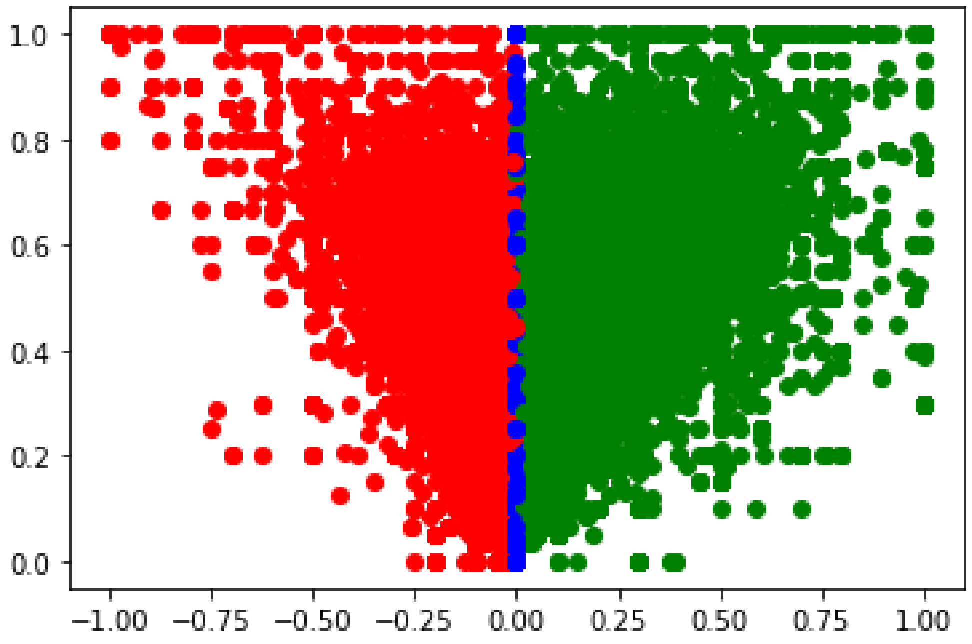


[23]:
```
for index, row in df.iterrows():
    if row['Score'] == 'Positive':
        plt.scatter(row['Polarity'], row['Subjectivity'], color="green")
    elif row['Score'] == 'Negative':
        plt.scatter(row['Polarity'], row['Subjectivity'], color="red")
    elif row['Score'] == 'Neutral':
        plt.scatter(row['Polarity'], row['Subjectivity'], color="blue")

plt.title('Twitter Sentiment Analysis')

plt.xlabel('Polarity')

plt.ylabel('Subjectivity')

plt.show()
```

```
     ␣
↪---------------------------------------------------------------------------

      KeyboardInterrupt                         Traceback (most recent call␣
↪last)

      <ipython-input-23-199bd91eb041> in <module>()
        3         plt.scatter(row['Polarity'], row['Subjectivity'],␣
↪color="green")
        4     elif row['Score'] == 'Negative':
  ----> 5         plt.scatter(row['Polarity'], row['Subjectivity'],␣
↪color="red")
        6     elif row['Score'] == 'Neutral':
        7         plt.scatter(row['Polarity'], row['Subjectivity'],␣
↪color="blue")


      /usr/local/lib/python3.7/dist-packages/matplotlib/pyplot.py in␣
↪scatter(x, y, s, c, marker, cmap, norm, vmin, vmax, alpha, linewidths, verts,␣
↪edgecolors, plotnonfinite, data, **kwargs)
     2814         verts=verts, edgecolors=edgecolors,
     2815         plotnonfinite=plotnonfinite, **({"data": data} if data is not
  -> 2816         None else {}), **kwargs)
     2817     sci(__ret)
     2818     return __ret


      /usr/local/lib/python3.7/dist-packages/matplotlib/__init__.py in␣
↪inner(ax, data, *args, **kwargs)
     1563     def inner(ax, *args, data=None, **kwargs):
     1564         if data is None:
  -> 1565             return func(ax, *map(sanitize_sequence, args), **kwargs)
     1566
     1567         bound = new_sig.bind(ax, *args, **kwargs)


      /usr/local/lib/python3.7/dist-packages/matplotlib/cbook/deprecation.py␣
↪in wrapper(*args, **kwargs)
      356                 f"%(removal)s.  If any parameter follows {name!r},␣
↪they "
      357                 f"should be pass as keyword, not positionally.")
  --> 358         return func(*args, **kwargs)
      359
      360     return wrapper
```

```
    /usr/local/lib/python3.7/dist-packages/matplotlib/axes/_axes.py in␣
↪scatter(self, x, y, s, c, marker, cmap, norm, vmin, vmax, alpha, linewidths,␣
↪verts, edgecolors, plotnonfinite, **kwargs)
   4464                 self.set_ymargin(0.05)
   4465
-> 4466         self.add_collection(collection)
   4467         self._request_autoscale_view()
   4468


    /usr/local/lib/python3.7/dist-packages/matplotlib/axes/_base.py in␣
↪add_collection(self, collection, autolim)
   1818             # Make sure viewLim is not stale (mostly to match
   1819             # pre-lazy-autoscale behavior, which is not really␣
↪better).
-> 1820             self._unstale_viewLim()
   1821             self.update_datalim(collection.get_datalim(self.
↪transData))
   1822


    /usr/local/lib/python3.7/dist-packages/matplotlib/axes/_base.py in␣
↪_unstale_viewLim(self)
    593             for ax in self._shared_y_axes.get_siblings(self):
    594                 ax._stale_viewlim_y = False
--> 595             self.autoscale_view(scalex=scalex, scaley=scaley)
    596
    597     @property


    /usr/local/lib/python3.7/dist-packages/matplotlib/axes/_base.py in␣
↪autoscale_view(self, tight, scalex, scaley)
   2388                 y_stickies = np.sort(np.concatenate([
   2389                     artist.sticky_edges.y
-> 2390                     for ax in self._shared_y_axes.get_siblings(self)
   2391                     if hasattr(ax, "lines")
   2392                     for artist in ax.get_children()]))


    /usr/local/lib/python3.7/dist-packages/matplotlib/axes/_base.py in␣
↪<listcomp>(.0)
   2390                     for ax in self._shared_y_axes.get_siblings(self)
   2391                     if hasattr(ax, "lines")
-> 2392                     for artist in ax.get_children()]))
   2393         if self.get_xscale().lower() == 'log':
   2394             x_stickies = x_stickies[x_stickies > 0]
```

```
    /usr/local/lib/python3.7/dist-packages/matplotlib/artist.py in␣
↪sticky_edges(self)
   1070
   1071         """
-> 1072         return self._sticky_edges
   1073
   1074     def update_from(self, other):


    KeyboardInterrupt:
```



[25]:
```python
objective = df[df['Subjectivity'] == 0]

print(str(objective.shape[0]/(df.shape[0])*100) + " % of objective tweets")
```

```
25.97903525888279 % of objective tweets
```

[26]:
```python
subjective = df[df['Subjectivity'] != 0]

print(str(subjective.shape[0]/(df.shape[0])*100) + " % of subjective tweets")
```

```
74.02096474111721 % of subjective tweets
```

Results from this section are:
Polarity

47.256886146027135 % of positive tweets

20.28406770431547 % of negative tweets

32.4590461496574 % of neutral tweets

Subjectivity

74.02096474111721 % of subjective tweets

25.97903525888279 % of objective tweets

7 5) Wordcloud creation and frequency analysis

These are the tasks we are going to perform here

1) Creating a word cloud
2) Hashtag analysis

    2.1) Total number of hashtags

    2.2) Total number of unique hashtags - Both case-sensitive and case-insensitive

    2.3) Sort hashtags based on frequency and compute top n hashtags

    2.4) Graphical representation

3) Twitter handle analysis

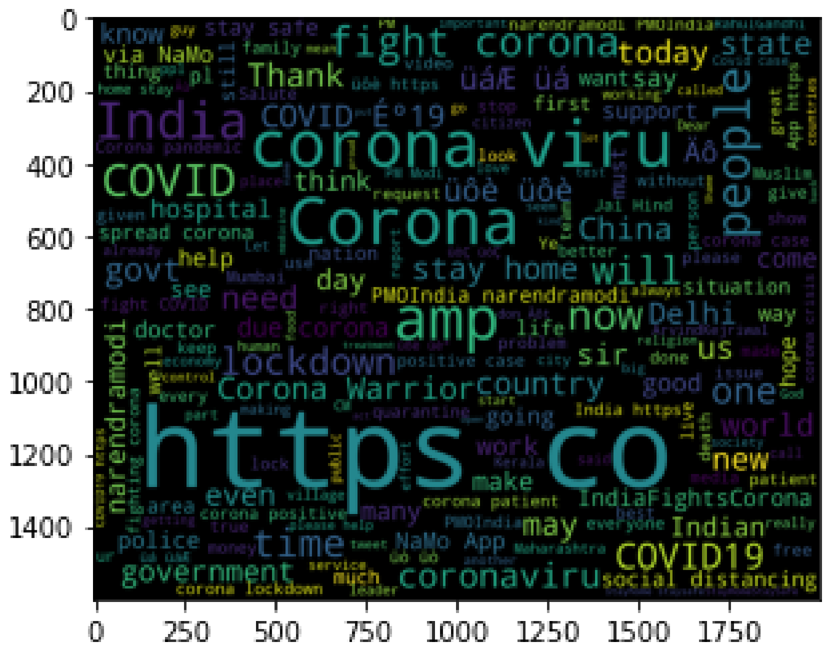
    3.1) Total number of handles

    3.2) Total number of unique handles - Both case-sensitive and case-insensitive

    3.3) Sort handles based on frequency and compute top n handles

    3.4) Graphical representation

[27]:
```
# Creating a word cloud

words = ' '.join([tweet for tweet in df['text']])

wordCloud = WordCloud(width=2000, height=1600).generate(words)

plt.imshow(wordCloud)

plt.show()
```



[28]:
```
def make_frequency_list(wordlist):
    freq = dict()

    for word in wordlist:
        if word in freq.keys():
            freq[word] += 1
        else:
            freq[word] = 1
    return freq
```

[29]:
```
# Separating out handles

handles_pattern = r'@\w+'

handles = nltk.regexp_tokenize(" ".join([word for word in df['text']]),␣
 ↪handles_pattern)

print("Total number of Twitter-handles in the tweets:", len(handles))

# Make a frequency list of handles(case sensitive)

handles_freq = make_frequency_list(handles)

print("Total number of Unique Twitter-handles(case sensitive):",␣
 ↪len(handles_freq.keys()))
```

```
# Make a frequency list of handles(case insensitive)

handles_case_insensitive = list(map(str.lower, handles))

handles_freq_in = make_frequency_list(handles_case_insensitive)

print("Total number of Unique tags(case insensitive):", len(handles_freq_in.
 ↪keys()))
```

```
Total number of Twitter-handles in the tweets: 59748
Total number of Unique Twitter-handles(case sensitive): 15453
Total number of Unique tags(case insensitive): 15334
```

[31]:
```
# Separating out hashtags

hashtags_pattern = r'#\w+'

hashtags = nltk.regexp_tokenize(" ".join([word for word in df['text']]),␣
 ↪hashtags_pattern)

print("Total number of hashtags in the tweets:", len(hashtags))

# Make a frequency list of hashtags(case sensitive)

hashtag_freq = make_frequency_list(hashtags)

print("Total number of Unique tags(case sensitive):", len(hashtag_freq.keys()))

# Make a frequency list of hashtags(case insensitive)

hashtags_case_insensitive = list(map(str.lower, hashtags))

hashtag_freq_in = make_frequency_list(hashtags_case_insensitive)

print("Total number of Unique tags(case insensitive):", len(hashtag_freq_in.
 ↪keys()))
```

```
Total number of hashtags in the tweets: 60539
Total number of Unique tags(case sensitive): 15092
Total number of Unique tags(case insensitive): 13220
```

[32]:
```
def sort_dictionary(dictionary, ascending=True):
    return {key: value for key, value in sorted(dictionary.items(), key=lambda␣
 ↪item: item[1], reverse=(not ascending))}
```

[33]:
```
def bar_plot(labels, values, title, xlabel, n):
    plt.title(title)
    plt.xlabel(xlabel)
    plt.ylabel('Frequency')
    for i in range(n):
        plt.text(i, values[i], str(values[i]))
    plt.xticks(range(n), labels=labels, rotation=90)
    plt.bar(range(n), height=values)
```

[34]:
```
# Top n hashtags

sorted_hashtags_freq = sort_dictionary(hashtag_freq, ascending=False)

@interact(n=(5, 50, 5))

def plot_histogram(n):
    labels = list(sorted_hashtags_freq.keys())[:n]
    values = list(sorted_hashtags_freq.values())[:n]
    bar_plot(labels, values, "Top "+str(n)+" trending hashtags on Twitter␣
 ↪during this Lockdown", "Hashtags", n)

    return
```

```
interactive(children=(IntSlider(value=25, description='n', max=50, min=5, step=5), Output()),
```

[35]:
```
# Top n handles

sorted_handles_freq = sort_dictionary(handles_freq, ascending=False)

@interact(n=(5, 50, 5))

def plot_histogram(n):
    labels = list(sorted_handles_freq.keys())[:n]
    values = list(sorted_handles_freq.values())[:n]
    bar_plot(labels, values, "Top "+str(n)+" trending handles on Twitter during␣
 ↪this Lockdown", "Handles", n)
    return
```

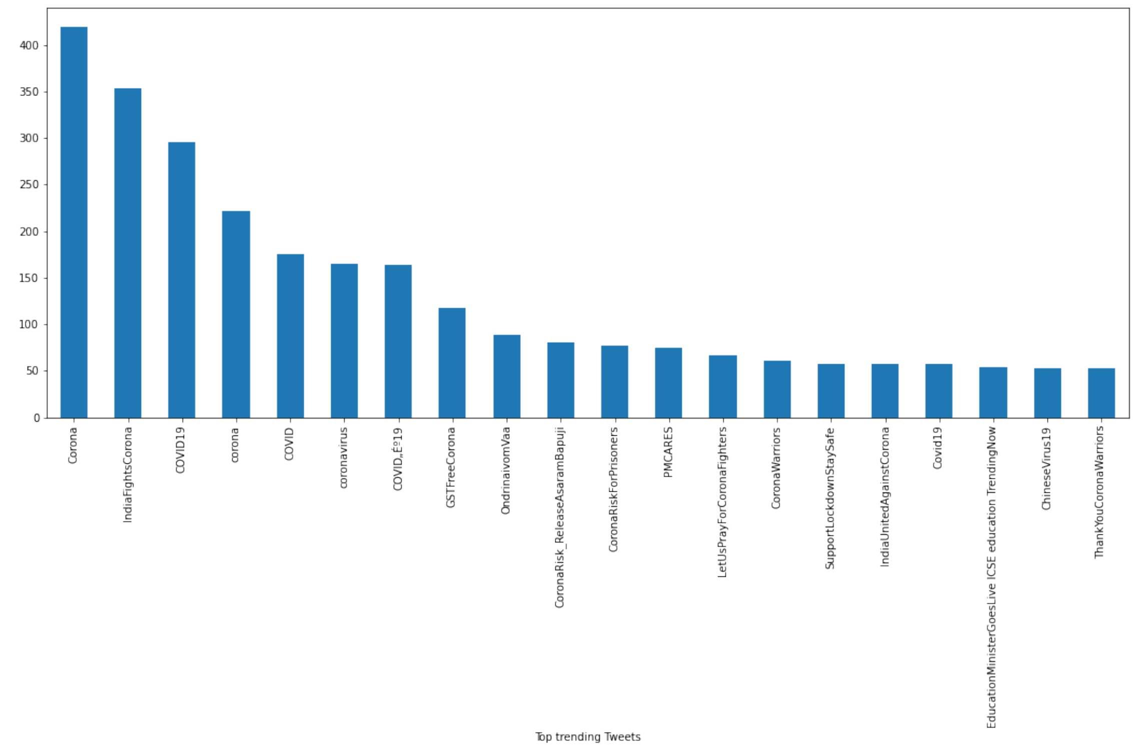
```
interactive(children=(IntSlider(value=25, description='n', max=50, min=5, step=5), Output()),
```

[36]:
```
print(df['hashtags'].value_counts().head(20))
```

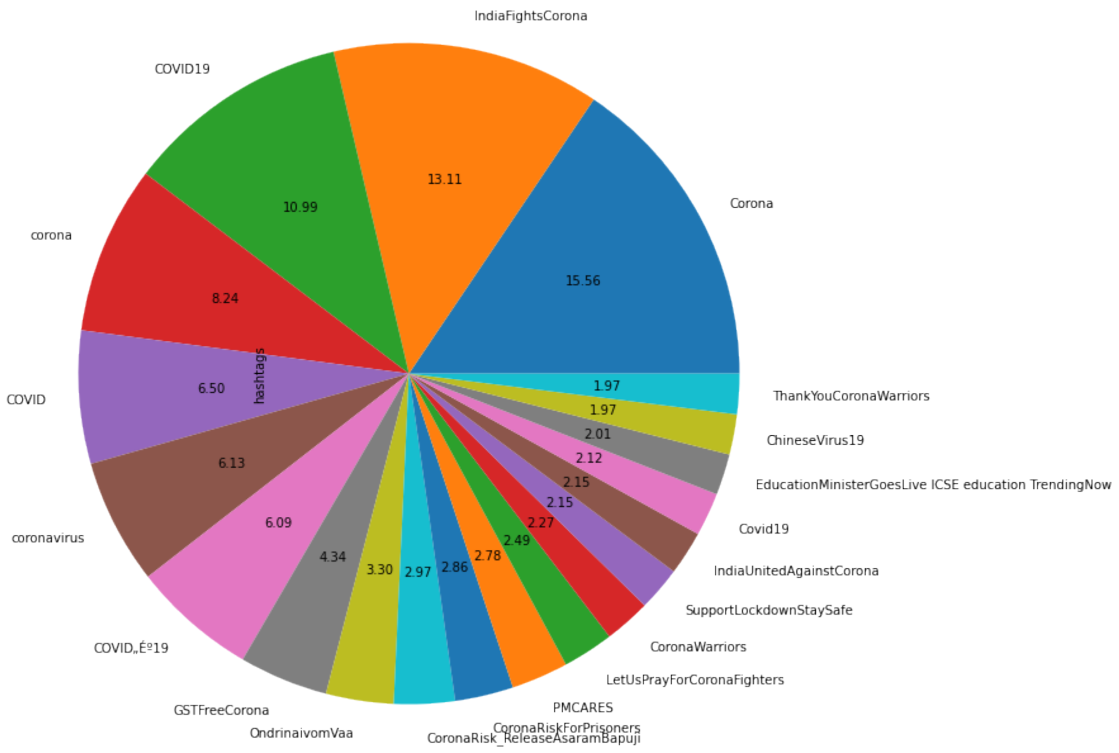
```
Corona                                                  419
IndiaFightsCorona                                       353
COVID19                                                 296
corona                                                  222
COVID                                                   175
```

```
coronavirus                                              165
COVIDÉž19                                               164
GSTFreeCorona                                            117
OndrinaivomVaa                                            89
CoronaRisk_ReleaseAsaramBapuji                            80
CoronaRiskForPrisoners                                    77
PMCARES                                                   75
LetUsPrayForCoronaFighters                                67
CoronaWarriors                                            61
SupportLockdownStaySafe                                   58
IndiaUnitedAgainstCorona                                  58
Covid19                                                   57
EducationMinisterGoesLive ICSE education TrendingNow      54
ChineseVirus19                                            53
ThankYouCoronaWarriors                                    53
Name: hashtags, dtype: int64
```

[37]:
```python
fig, ax = plt.subplots()

plt.xlabel('Top trending Tweets')

df['hashtags'].value_counts().head(20).plot(ax=ax, kind='bar',figsize=(18,7));
```



```
[38]: df['hashtags'].value_counts().head(20).plot(kind='pie', autopct='%.2f',␣
      ↪radius=3);
```



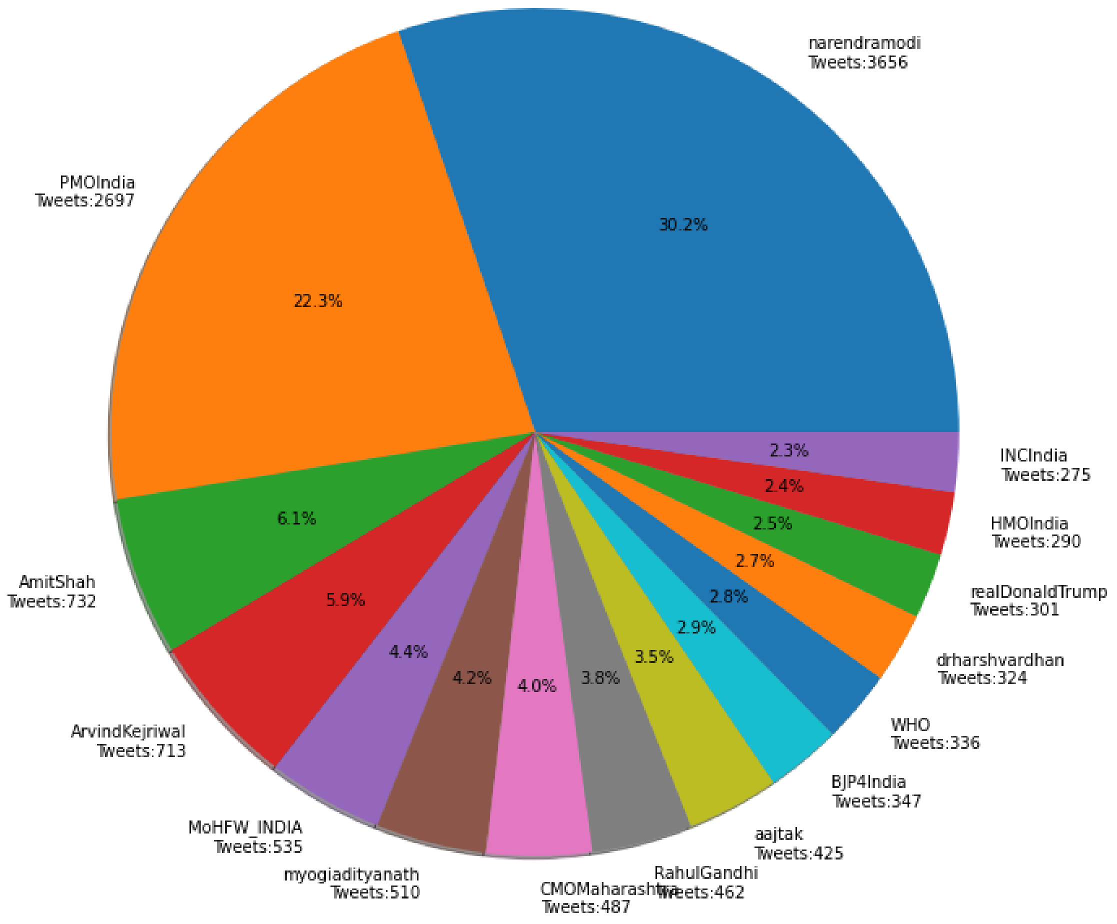
```
[39]: raw = ' '.join([word for word in df['text']])

      tags = [re.sub(r"(\W+)$", "", j[1:]) for j in [i for i in raw.split() if i.
      ↪startswith("@") and len(i) != 1 ]]

      df1 = pd.DataFrame({"handlers": tags})
```

```
[40]: df1_count= df1['handlers'].value_counts().head(15)

      print(df1_count)
```

```
narendramodi       3656
PMOIndia           2697
AmitShah            732
ArvindKejriwal      713
MoHFW_INDIA         535
myogiadityanath     510
CMOMaharashtra      487
RahulGandhi         462
aajtak              425
BJP4India           347
```

```
WHO                   336
drharshvardhan        324
realDonaldTrump       301
HMOIndia              290
INCIndia              275
Name: handlers, dtype: int64
```

[41]:
```
def add(a, b):
    return a+'\nTweets:'+str(b)

plt.pie(df1_count, autopct='%1.1f%%', shadow=True,  radius=3,␣
 ↪labels=list(map(add,df1_count.index.values, df1_count)));

plt.show();
```



Results from this section are:
Twitter handles

Total number of Twitter-handles in the tweets: 59748

Total number of Unique Twitter-handles(case sensitive): 15453

Total number of Unique tags(case insensitive): 15334

Top 5 handles

@narendramodi

@PMOIndia

AmitShah

@ArvindKejriwal

@MoHFW_INDIA

Hashtags

Total number of hashtags in the tweets: 60539

Total number of Unique tags(case sensitive): 15092

Total number of Unique tags(case insensitive): 13220

Top 5 hashtags

Corona

IndiaFightsCorona

COVID19

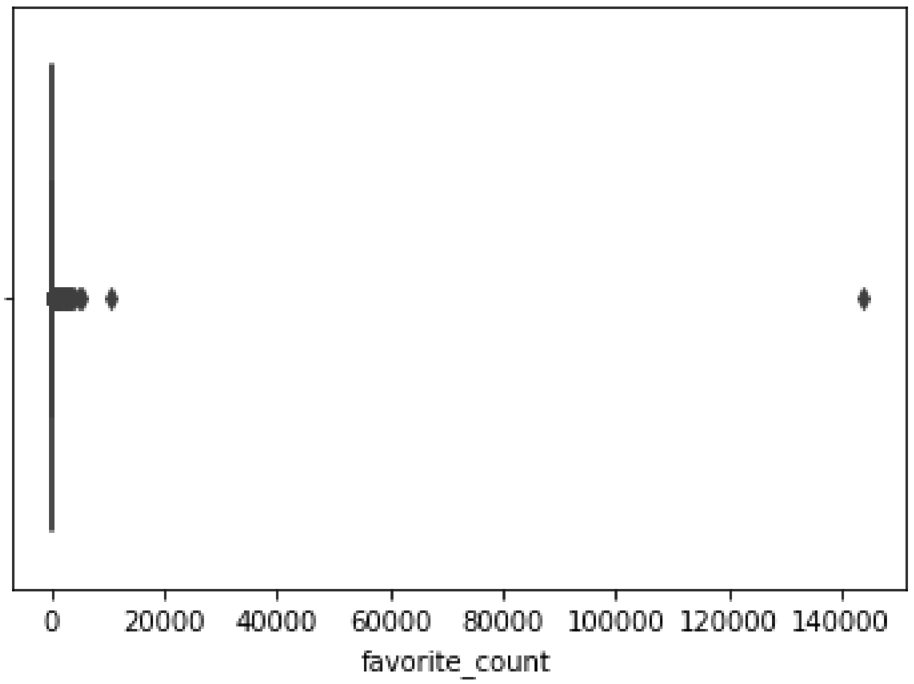
corona

COVID

8 6) Different plots

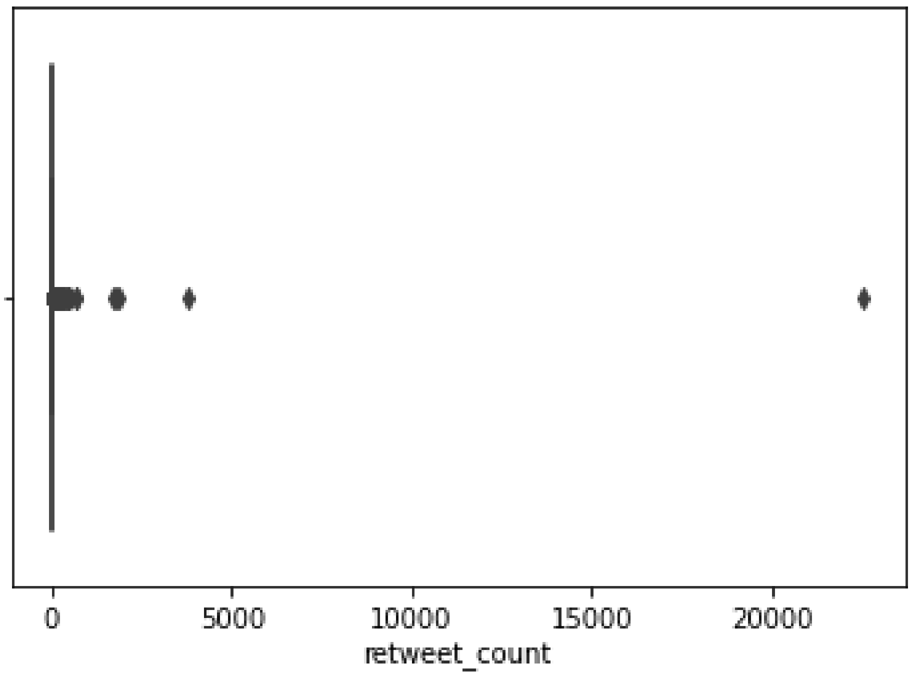
[42]:
```
sns.boxplot(x=df['favorite_count'])
```

[42]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcee52281d0>
```



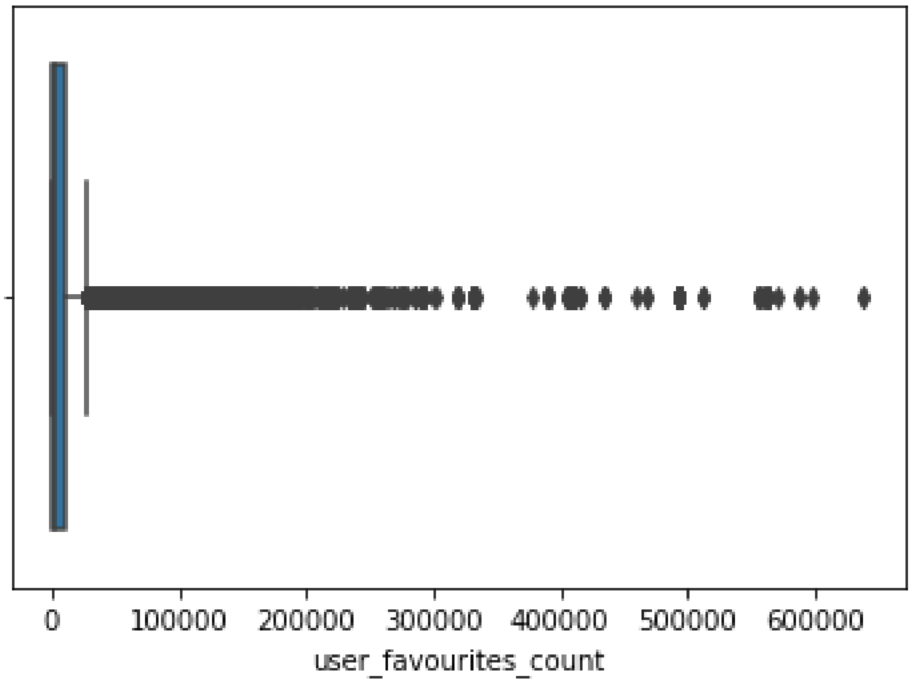
[43]:
```
sns.boxplot(x=df['retweet_count'])
```

[43]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fceda783a90>
```



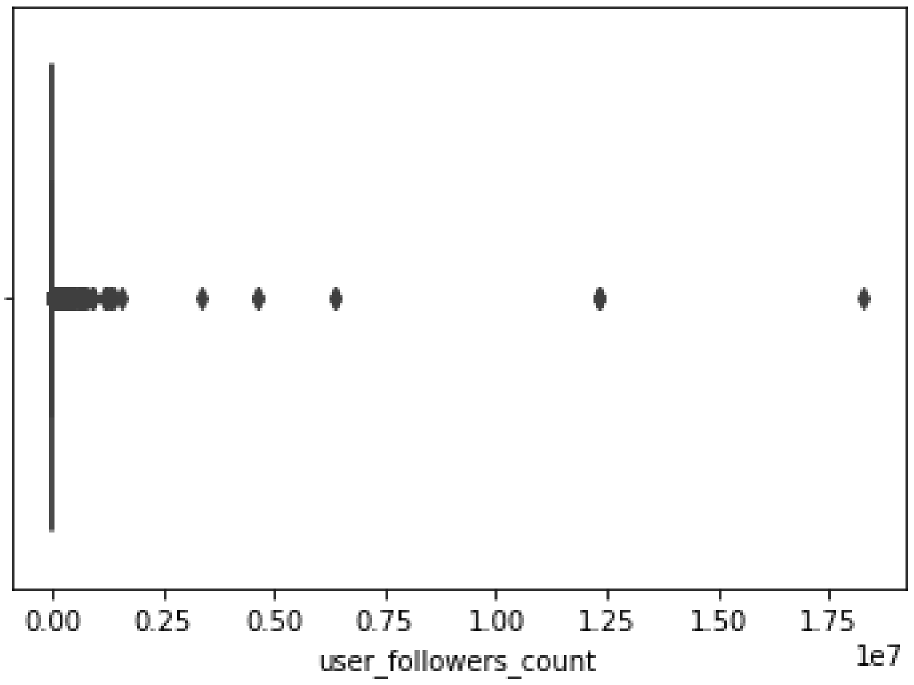
[44]:
```
sns.boxplot(x=df['user_favourites_count'])
```

[44]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcee5ed04d0>
```



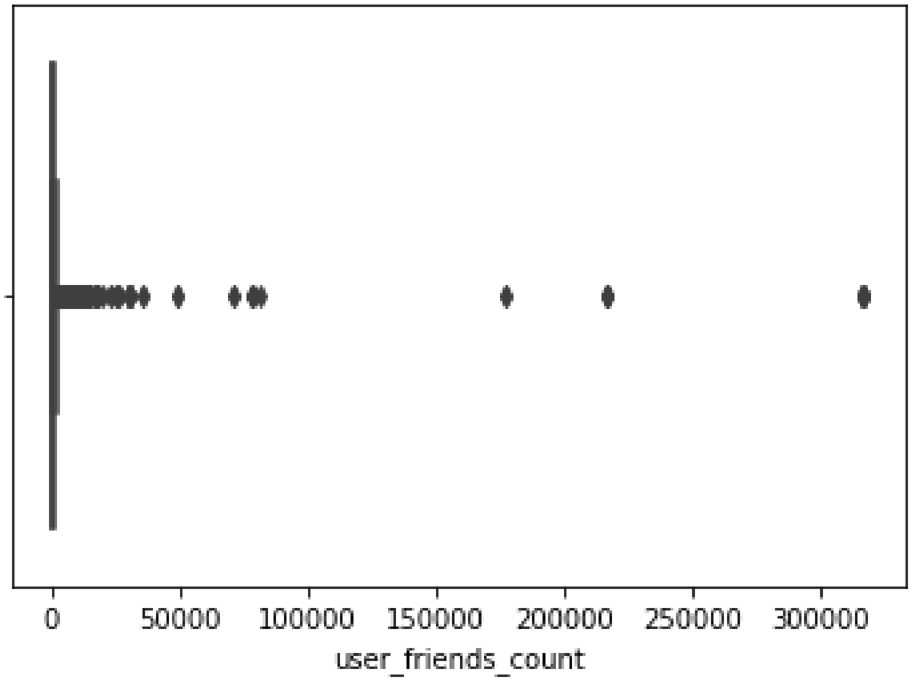
[45]:
```
sns.boxplot(x=df['user_followers_count'])
```

[45]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcee5d3f450>
```



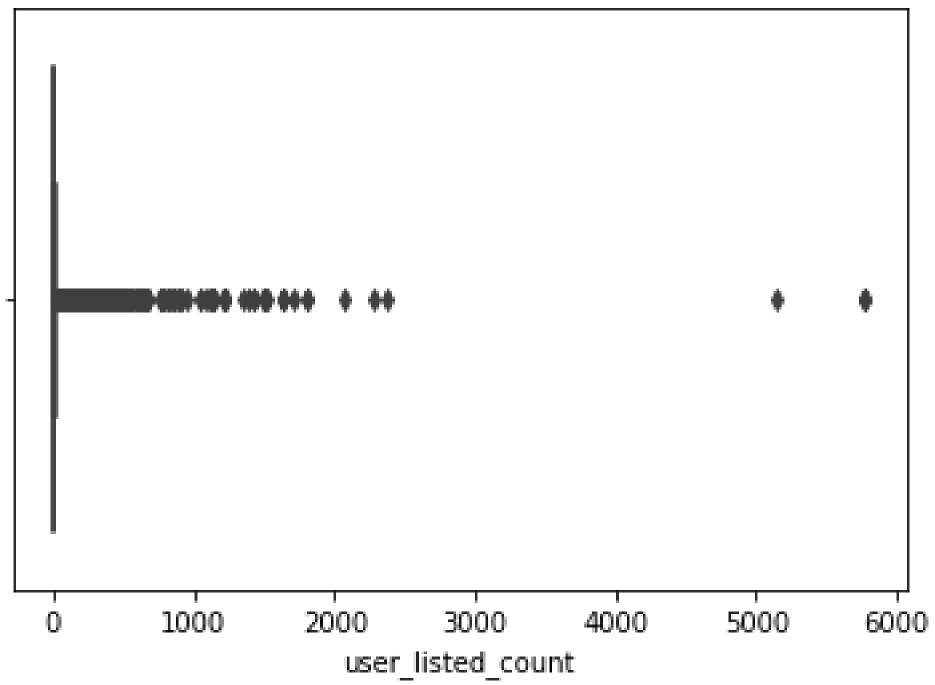
[46]:
```
sns.boxplot(x=df['user_friends_count'])
```

[46]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcee5bfb450>
```



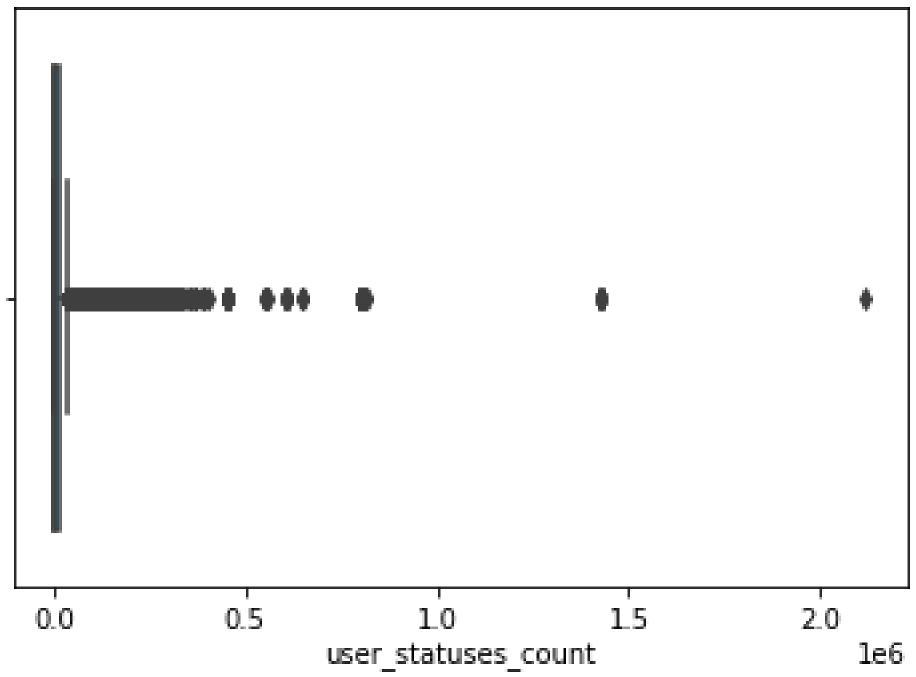
[47]:
```
sns.boxplot(x=df['user_listed_count'])
```

[47]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcee5c576d0>
```

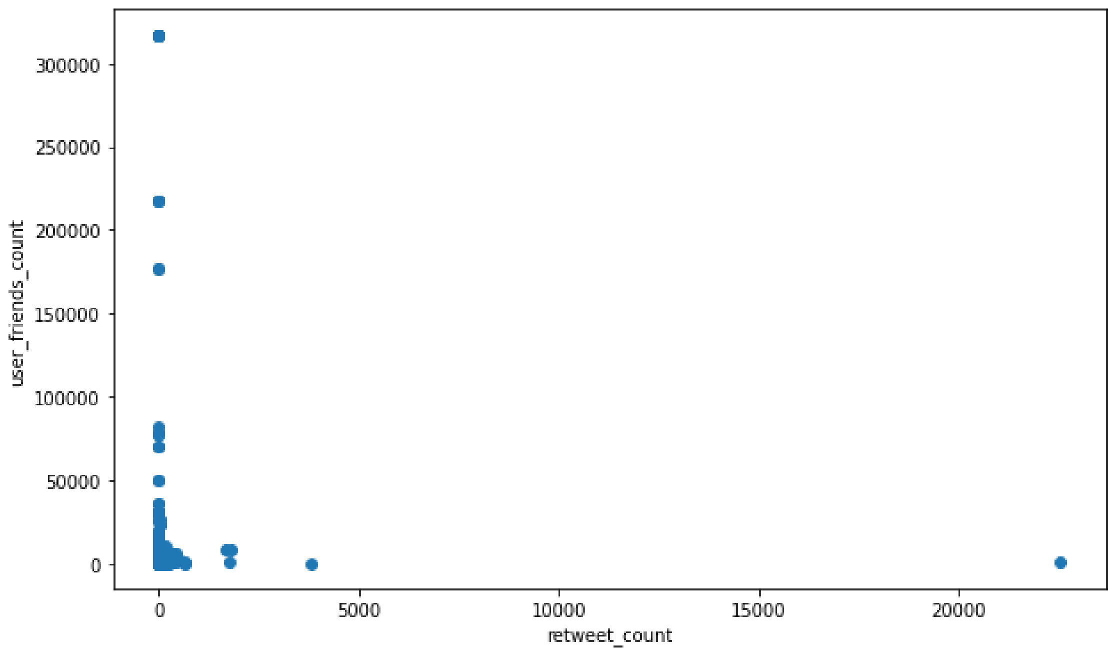


[48]:
```
sns.boxplot(x=df['user_statuses_count'])
```

[48]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcee5b14dd0>
```



[49]:
```
fig, ax = plt.subplots(figsize=(10,6))

ax.scatter(df['retweet_count'], df['user_friends_count'])

ax.set_xlabel('retweet_count')

ax.set_ylabel('user_friends_count')

plt.show()
```



9 7) Conclusion

The Covid infection keeps on spreading across the world after a direction that is hard to anticipate. The wellbeing, helpful and financial approaches received by nations will decide the speed and strength of the recuperation.

A planned worldwide exertion is needed to help nations that presently don't have adequate monetary space to fund social approach, specifically, widespread social insurance frameworks

Not every person was ready for the trial of the pandemic. Much under the current conditions, when worldwide difficulties should join individuals and impel individuals to even briefly fail to remember divergences, some actually resort to abuse. Not every person can oppose the enticement of being narrow minded. Others additionally exploit the circumstance to play international affairs by pursuing their own advantages and retribution against their international adversaries. Once reared in such a climate, the infection will strengthen clashes and increase uncalled for rivalry.

It's an ideal opportunity to surrender regular intuition dependent on generalizations, and begin acting from an ethical viewpoint. All things considered, our smartest option is a cheerful future for all who live on Earth, our regular home.

10 *Let us all pray for such a time to come as soon as possible, and also for a better Earth to live in.*

10.1 Rohit Narayanan

10.2 National Institute of Technology Puducherry

Email:rohitnarayanan1729@gmail.com

[ ]:
```
!wget -nc https://raw.githubusercontent.com/brpy/colab-pdf/master/colab_pdf.py
from colab_pdf import colab_pdf
colab_pdf('SpotleCovidAnalysis.ipynb')
```

```
--2021-06-06 07:50:51--  https://raw.githubusercontent.com/brpy/colab-
pdf/master/colab_pdf.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)...
185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com
(raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1864 (1.8K) [text/plain]
Saving to: colab_pdf.py

colab_pdf.py        100%[===================>]   1.82K  --.-KB/s    in 0s

2021-06-06 07:50:51 (21.3 MB/s) - colab_pdf.py saved [1864/1864]

Mounted at /content/drive/

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.


WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Extracting templates from packages: 100%
```