Twitter

June 25, 2021

1 Data Science: Twitter data analyis

[30]:
```
# library imports
import json, gzip
import pandas as pd
import sqlite3
import os
import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime
import folium
from folium import plugins
from wordcloud import WordCloud, STOPWORDS
```

1.1 Processing the data.

Out of all the data available in JSON form, 32 separate db tables were created. 31 tables are maintaining the tweets related data each containing the tweet data for a day of March month. They have attributes like tweet_id, user_id, tweet_text, created_at, user_mentions, boud_lat, boundlong etc. These attributes are important enough to do all the required data analysis to answer the questions. The primary key for these 31 tables is tweet_id which is unique. One table is maintained for user details like user_id and user_name. Each table has tweet data of a day of March month that is relevant to complete all the tasks. In addition, a user table is also created to maintain user data. SQLITE database is used for storing this data.

1.2 Creating a user table

[32]:
```
# creating a user table
conn = sqlite3.connect('tweets2.sqlite')
c = conn.cursor()
c.execute('''CREATE TABLE {}
             (user_id int, user_name text )'''.format('users'))
conn.commit()
```

[ ]:
```
# this function reads the data from json files and put it in a separate SQLITE␣
↪database table for each day
def process_tweet_data(file, db_cursor, conn):
```

```
    with gzip.open(file, 'r') as f:
        lines = f.readlines()
        no_of_lines = len(lines)
        print(no_of_lines)
        # Create table
        table_name = 'tweets_'+file.split('_')[1].split('.')[0]
        db_cursor.execute('''CREATE TABLE {}
                    (tweet_id int, user_id int, lang text, source text,␣
↪tweet_text text, created_at text,
                    bound_lat real, bound_long real, bound_country text, lat␣
↪real, long real, user_mentions text  )'''
                  .format(table_name))
        rows = []
        user_rows = []
        row_count = 0
        for n in range(no_of_lines):
            line = lines[n]
            data = json.loads(line)
            # tweet data
            try:
                tweet_id = data['id']
                user_id = data['user']['id']
                lang = data['lang']
                source = data['source']
                tweet_text = data['text']
                bound_lat = bound_long = bound_country = lat = long = None
                if data['place'] is not None:
                    if data['place']['bounding_box'] is not None:
                        bound_lat =␣
↪data['place']['bounding_box']['coordinates'][0][0][1]
                        bound_long =␣
↪data['place']['bounding_box']['coordinates'][0][0][0]
                        bound_country = data['place']['country_code']
                if data['coordinates'] is not None:
                    lat = data['coordinates']['coordinates'][1]
                    long = data['coordinates']['coordinates'][0]
                created_at = data['timestamp_ms']
                user_mentions = data['entities']['user_mentions']
                # updating users table with user mention
                for item in user_mentions:
                    user_rows.append((item['id'], item['name']))
                user_mentions = str([item['id'] for item in user_mentions])

                rows.append((tweet_id, user_id, lang, source, tweet_text,␣
↪created_at, bound_lat, bound_long, bound_country,
                             lat, long, user_mentions))
                # user data
```

```
                user_id = data['user']['id']
                user_name = data['user']['name']
                user_rows.append((user_id, user_name))
                row_count += 1
                # Insert 1000 rows of data into the table
                if row_count == 1000 or n==no_of_lines-1:
                    db_cursor.executemany("INSERT INTO {} VALUES (?, ?, ?, ?, ?
↪, ?, ?, ?, ?, ?, ?, ?)".format(table_name),rows)
                    db_cursor.executemany("INSERT INTO {} VALUES (?, ?)".
↪format('users'),user_rows)
                    conn.commit()
                    row_count = 0
                    rows = []
                    user_rows = []
            except KeyError:
                continue
```

[ ]:

```
# calling the above function on all gz files to store data in 32 db tables.
path = os.getcwd()+'\\Twitter Data'
gz_files = os.listdir(path)
#gz_files = ['\\geoEurope_20200329.gz']
for file in gz_files:
    process_tweet_data(path+'\\' + file, db_cursor=c, conn=conn)
    print('{} done'.format(file))
conn.close()
```

Because of the way, user entries were made to the users table, there would be some duplicate entries of users. Following sql query is used to remove the duplicate users.

[ ]:

```
# SQL query for removing the duplicate entries in the users table
"""
DELETE FROM users
WHERE rowid NOT IN (
  SELECT MIN(rowid)
  FROM users
  GROUP BY user_id
)
"""
```

1.3 Code or function definitions used in answering Part 1.

[4]:

```
# Function to get the tweet count by day
def get_tweet_count_by_day():
    conn = sqlite3.connect('tweets2.sqlite')
    c = conn.cursor()
    all_tables = c.execute("SELECT name FROM sqlite_master  WHERE type='table'␣
↪and name like 'tweets%';")
```

```
    tweet_count_by_day = {}
    for table in all_tables.fetchall():
        table_name = table[0]
        count = c.execute("SELECT count(*) from {}".format(table_name))
        count = count.fetchone()[0]
        day_timestamp = c.execute("SELECT created_at from {} limit 1".
 ↪format(table_name))
        day_timestamp = day_timestamp.fetchone()[0]
        tweet_count_by_day.update({day_timestamp:[count]})
    conn.close()
    df_tweets_by_day = pd.DataFrame.from_dict(tweet_count_by_day,␣
 ↪orient='index', columns= ['count'])
    df_tweets_by_day['weekday'] =  df_tweets_by_day.index
    df_tweets_by_day['date'] = df_tweets_by_day['weekday'].apply(lambda a :␣
 ↪datetime.fromtimestamp(int(a)/1000).date())
    df_tweets_by_day['weekday'] = df_tweets_by_day['weekday'].apply(lambda a :␣
 ↪datetime.fromtimestamp(int(a)/1000).weekday())
    return df_tweets_by_day

def plot_no_of_tweets_by_day(df_tweets_by_day):
    # Make a bar chart of number of tweets by day
    fig, axes = plt.subplots(nrows=1, ncols=1, figsize=(12,4), dpi =300)
    axes.bar(df_tweets_by_day['date'].apply(lambda a: a.day),␣
 ↪df_tweets_by_day['count'])
    axes.set_xlabel('A day of March month')
    axes.set_ylabel('No of tweets')
    axes.set_title('Number of tweets by day')
    fig.tight_layout()
    plt.show()

# Get the tweet count by language
def get_tweet_count_by_language():
    """return pandas series"""
    conn = sqlite3.connect('tweets2.sqlite')
    c = conn.cursor()
    all_tables = c.execute("SELECT name FROM sqlite_master  WHERE type='table'␣
 ↪and name like 'tweets%';")
    ser = pd.Series([], dtype=np.int64)
    for table in all_tables.fetchall():
        table_name = table[0]
        ser1 = pd.read_sql_query("SELECT lang from {}".format(table_name), conn)
        ser1 = ser1['lang'].value_counts()
        ser = ser.add(ser1, fill_value=0).astype(int)
    conn.close()
    return ser
```

```
def plot_no_of_tweets_by_language(series):
    series = series[series>=100000]
    # Make a bar chart of number of tweets by language
    fig, axes = plt.subplots(nrows=1, ncols=1, figsize=(12,4), dpi =300)
    axes.bar(series.index, series)
    axes.set_xlabel('Language')
    axes.set_ylabel('No of tweets')
    axes.set_title('Number of tweets by language')
    fig.tight_layout()
    plt.show()

# plotting box and whisker plot
def box_plot_weekend_to_weekday(data):
    plt.boxplot(data)
    plt.xticks([1, 2], ['Weekday','Weekend'])
    #plt.ylim(400000,990000)
    plt.title('Weekend & Weekday Tweet Comparision')
    plt.ylabel('No Of Tweets')
    plt.show()

    # Get the tweet count by hour for 31 days of March
def get_tweet_count_by_hour():
    conn = sqlite3.connect('tweets2.sqlite')
    c = conn.cursor()
    all_tables = c.execute("SELECT name FROM sqlite_master  WHERE type='table'␣
 ↪and name like 'tweets%';")
    ser = pd.Series([], dtype=np.int64)
    for table in all_tables.fetchall():
        table_name = table[0]
        ser1 = pd.read_sql_query("SELECT created_at from {}".
 ↪format(table_name), conn)
        ser1['hour'] = ser1['created_at'].apply(lambda a: datetime.
 ↪fromtimestamp(int(a)/1000).time().hour)
        ser1 = ser1['hour'].value_counts()
        ser = ser.add(ser1, fill_value=0).astype(int)
    conn.close()
    return ser

def plot_tweet_count_by_hour(series):
    # Make a bar chart of number of tweets by hour
    fig, axes = plt.subplots(nrows=1, ncols=1, figsize=(12,4), dpi =300)
    axes.bar(series.index, series)
    axes.set_xlabel('Hour')
    axes.set_ylabel('Average number of tweets')
    axes.set_title('Average number of tweets by hour for March month in Europe')
    fig.tight_layout()
    plt.show()
```

1.4 Code or function definitions used in answering Part 2.

[34]:
```
# Get the coordinates of tweets for 31 days
def get_tweet_coordinates():
    conn = sqlite3.connect('tweets2.sqlite')
    c = conn.cursor()
    all_tables = c.execute("SELECT name FROM sqlite_master  WHERE type='table'␣
 ↪and name like 'tweets%';")
    df_coordinates = pd.DataFrame([],columns=['lat', 'long'])
    for table in all_tables.fetchall():
        table_name = table[0]
        df = pd.read_sql_query("SELECT lat, long from {}".format(table_name),␣
 ↪conn)
        df.dropna(inplace=True)
        df_coordinates = df_coordinates.append(df, ignore_index=True)
    conn.close()
    return df_coordinates

def plot_map(df_coordinates):
    heatmap = folium.Map(location=[52.5200,13.4050], zoom_start=3,␣
 ↪control_scale = True)
    data = [(row['lat'],row['long']) for index, row in df_coordinates.
 ↪iterrows()]
    heatmap.add_child(plugins.HeatMap(data=data, radius=10))
    heatmap.add_child(plugins.MarkerCluster(data=data))
    heatmap.save("./folium_heatmap.html")

    return heatmap
```

1.5 Code or function definitions used in answering Part 3.

[6]:
```
def get_no_of_tweets_per_user():
    # getting the no of tweets per user
    conn = sqlite3.connect('tweets2.sqlite')
    c = conn.cursor()
    all_tables = c.execute("SELECT name FROM sqlite_master  WHERE type='table'␣
 ↪and name like 'tweets%';")
    ser = pd.Series([], dtype=np.int64)
    for table in all_tables.fetchall():
        table_name = table[0]
        ser1 = (pd.read_sql_query("SELECT user_id from {}".format(table_name),␣
 ↪conn))
        ser1 = ser1['user_id'].value_counts()
        ser = ser.add(ser1, fill_value=0).astype(int)
    conn.close()
    return ser
```

```python
def plot_no_of_tweets_per_user(ser):
    ser = ser.value_counts()
    # Make a histogram of number of tweets per user
    fig, axes = plt.subplots(nrows=1, ncols=1, figsize=(12,4), dpi =300)
    axes.bar(ser.index, ser)
    axes.set_xlabel('No of users')
    axes.set_ylabel('No of tweets')
    axes.set_title('Number of tweets per user')
    #axes.set_ylim(0,60000)
    axes.set_xlim(0,100)
    fig.tight_layout()
    plt.show()

def get_usernames(userids):
    conn = sqlite3.connect('tweets2.sqlite')
    c = conn.cursor()
    df = pd.read_sql_query("SELECT user_id, user_name from users", conn)
    conn.close()
    user_names = []
    for id in userids:
        user_names.append(df.loc[:,'user_name'][df['user_id']==id].values[0])
    return user_names


# getting the no of mentions for a user
def get_no_of_mentions_for_a_user():
    conn = sqlite3.connect('tweets2.sqlite')
    c = conn.cursor()
    all_tables = c.execute("SELECT name FROM sqlite_master  WHERE type='table'␣
↪and name like 'tweets%';")
    a = np.array([], dtype=np.int64)
    for table in all_tables.fetchall():
        table_name = table[0]
        df = pd.read_sql_query("SELECT user_mentions from {}".
↪format(table_name), conn)
        df[df['user_mentions']=='[]'] = None
        df.dropna(inplace=True)
        df.index= range(len(df))
        list1 = []
        for i in range(len(df)):
            item = df.loc[i, 'user_mentions']
            list1 += eval(item)
        a = np.append(a, np.array(list1, dtype=np.int64), axis=0)
    conn.close()
    users_mentioned = pd.Series(a, dtype=np.int64).value_counts()
    return users_mentioned
```

```python
def plot_no_of_mentions_per_user(series):
    series = series.value_counts().sort_index()
    # Make a histogram of number of mentions per user
    fig, axes = plt.subplots(nrows=1, ncols=1, figsize=(12,4), dpi =300)
    axes.bar(series.index, series)
    axes.set_xlabel('No of users')
    axes.set_ylabel('No of mentions')
    axes.set_title('Number of mentions per user')
    #axes.set_ylim(0,60000)
    axes.set_xlim(0,50)
    fig.tight_layout()
    plt.show()
```

1.6 Code or function definitions used in answering Part 4.

[4]:
```python
def get_tweet_count_by(country, day):
    conn = sqlite3.connect('tweets2.sqlite')
    c = conn.cursor()
    table_name = 'tweets_2020'+day
    df = pd.read_sql_query("SELECT bound_country from {}".format(table_name),␣
 ↪conn)
    count = np.sum(df['bound_country']==country, axis=0)
    conn.close()
    return count

def get_tweet_trends_for_country(country):
    counts = [get_tweet_count_by(country,'030'+str(i)) if i<10 else␣
 ↪get_tweet_count_by(country,'03'+str(i))
              for i in range(1,32)]
    return counts

def plot_tweets_vs_day_for_country(country, y):
    # Make a plot of number of mentions per user
    fig, axes = plt.subplots(nrows=1, ncols=1, figsize=(12,4), dpi =300)
    axes.bar(range(1,len(y)+1), y)
    axes.set_xlabel('Day of March month')
    axes.set_ylabel('No of tweets ({})'.format(country))
    axes.set_title('No of tweets on a day of March in {}'.format(country))
    fig.tight_layout()
    plt.show()

def select_hashtag_words(text):
    output = ''
    for word in text.split(' '):
        if word!= '' and word[0] == '#':
            output += ' '+word
    output = output.replace('#', '')
```

```
    return output

def get_tweet_trends(country, day):
    conn = sqlite3.connect('tweets2.sqlite')
    c = conn.cursor()
    table_name = 'tweets_2020'+day
    df = pd.read_sql_query("SELECT tweet_text from {} where bound_country =␣
 ↪'{}'".format(table_name, country), conn)
    df['trend_words'] = df['tweet_text'].apply(lambda a:␣
 ↪select_hashtag_words(a))
    trend_words = df['trend_words'].str.cat(sep=' ')
    conn.close()
    return trend_words


def plot_word_cloud(trend_words):
        stopwords = set(STOPWORDS)
        wordcloud = WordCloud(width = 800, height = 800, background_color␣
 ↪='white', stopwords = stopwords, min_font_size = 10)
        wordcloud.generate(trend_words)
        # plot the WordCloud image
        plt.figure(figsize = (8, 8), facecolor = None)
        plt.imshow(wordcloud)
        plt.axis("off")
        plt.tight_layout(pad = 0)
        plt.show()
```

2 Part 1. Basic Stats

1. Download the data.
2. Count the total number of tweets, describing how you deal with duplicates or other anomalies in the data set.
3. Plot a time-series of the number of tweets by day. Comment on what you see.
4. Show a breakdown of the number of tweets by language and discuss your findings
5. Using a box and whisker diagram, compare the number of tweets on a weekday (Monday-Friday) to weekend days (Saturday-Sunday). Observe the pattern.
6. There are multiple different time zones across Europe. Accounting for this, plot the average number of tweets at each hour of the day across the time period.

2.0.1 Dealing with duplicates or other anomalies in the data set

While performing the data processing and storing the data into SQLITE tables, tweets that were empty or had most of the null attributes were skipped. Even after the data processing, there were some duplicates of tweets in the data. These duplicate instances can be easily identified by the tweet_id in the tweet tables and hence, can easily be removed by an sql query. The tweet_id is a unique attribute. The SQL query is given below.

[8]:
```python
# Removing the duplicate tweets from all the 31 tables by sql query.
conn = sqlite3.connect('tweets2.sqlite')
c = conn.cursor()
all_tables = c.execute("SELECT name FROM sqlite_master  WHERE type='table' and␣
 ↪name like 'tweets%';")

for table in all_tables.fetchall():
    table_name = table[0]
    c.execute('''DELETE FROM {}
                WHERE rowid NOT IN (
                  SELECT MIN(rowid)
                  FROM {}
                  GROUP BY tweet_id
                )'''.format(table_name,table_name))
conn.commit()
conn.close()
```

2.0.2 Counting the total number of tweets

[27]:
```python
def get_tweet_count_by_day():
    conn = sqlite3.connect('tweets2.sqlite')
    c = conn.cursor()
    all_tables = c.execute("SELECT name FROM sqlite_master  WHERE type='table'␣
 ↪and name like 'tweets%';")
    tweet_count_by_day = {}
    for table in all_tables.fetchall():
        table_name = table[0]
        count = c.execute("SELECT count(*) from {}".format(table_name))
        count = count.fetchone()[0]
        day_timestamp = c.execute("SELECT created_at from {} limit 1".
 ↪format(table_name))
        day_timestamp = day_timestamp.fetchone()[0]
        tweet_count_by_day.update({day_timestamp:[count]})
    conn.close()
    df_tweets_by_day = pd.DataFrame.from_dict(tweet_count_by_day,␣
 ↪orient='index', columns= ['count'])
    df_tweets_by_day['weekday'] =  df_tweets_by_day.index
    df_tweets_by_day['date'] = df_tweets_by_day['weekday'].apply(lambda a :␣
 ↪datetime.fromtimestamp(int(a)/1000).date())
    df_tweets_by_day['weekday'] = df_tweets_by_day['weekday'].apply(lambda a :␣
 ↪datetime.fromtimestamp(int(a)/1000).weekday())
    return df_tweets_by_day
```

The number of tweets by day can easily be counted using the function defined above.It returns a dataframe of day wise tweets, which can be used to answer further questions. The total number of tweets in March month of 2020 in Europe were 24794903.

[9]:
```
# getting the day wise tweet count. The table below gives the daywise count of␣
↪tweets in March.
df_tweets_by_day = get_tweet_count_by_day()
df_tweets_by_day
```

[9]:

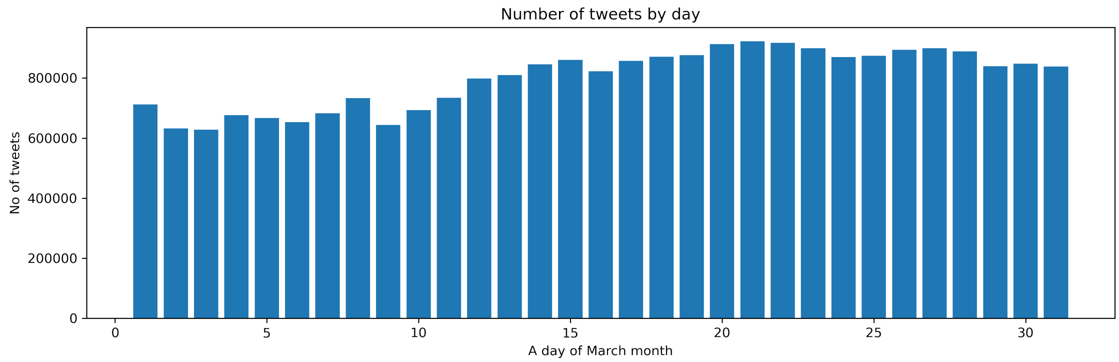
	count	weekday	date
1583020800156	711695	6	2020-03-01
1583107200014	632341	0	2020-03-02
1583193600125	627611	1	2020-03-03
1583280000020	675752	2	2020-03-04
1583366400150	667126	3	2020-03-05
1583452800072	653005	4	2020-03-06
1583539200078	682269	5	2020-03-07
1583625600069	732684	6	2020-03-08
1583712000226	644021	0	2020-03-09
1583798400007	692675	1	2020-03-10
1583884800063	734144	2	2020-03-11
1583971200191	798491	3	2020-03-12
1584057600142	809308	4	2020-03-13
1584144000142	845711	5	2020-03-14
1584230400082	860722	6	2020-03-15
1584316800171	822663	0	2020-03-16
1584403200345	857119	1	2020-03-17
1584489600108	870964	2	2020-03-18
1584576000044	875599	3	2020-03-19
1584662400002	913107	4	2020-03-20
1584748800002	922452	5	2020-03-21
1584835200134	917071	6	2020-03-22
1584921600166	898977	0	2020-03-23
1585008000131	869536	1	2020-03-24
1585094400142	873908	2	2020-03-25
1585180800052	893533	3	2020-03-26
1585267200190	899658	4	2020-03-27
1585353600147	888549	5	2020-03-28
1585440000149	838827	6	2020-03-29
1585522800161	847671	0	2020-03-30
1585609200010	837714	1	2020-03-31

[10]:
```
# Counting the total number of tweets
np.sum(df_tweets_by_day['count'])
```

[10]: 24794903

2.0.3 Timeseries plot of the number of tweets by day

[12]:
```
# Plotting the no of tweets by day
plot_no_of_tweets_by_day(df_tweets_by_day)
```



2.0.4 Observation

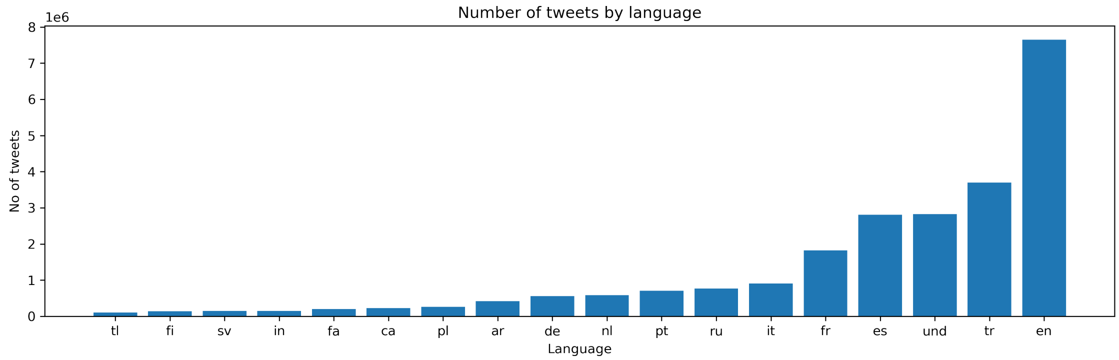
Overall,there is an increasing trend in the number of tweets for the whole month.The month started with lesser number of day tweets but after 11th march, more number of tweets were made daily. Probably, this may be accounted towards the onset of COVID virus pandemic. Apart from this, there is one more trend visible. On 6th and 7th days of the week (saturday and sunday), there are always more number of tweets compared to the weekdays.

2.0.5 Breakdown of the number of tweets by language

[13]:
```
# getting the breakdown of tweet couny by language
tweet_count_by_language = get_tweet_count_by_language()
tweet_count_by_language = tweet_count_by_language.sort_values()
tweet_count_by_language
```

[13]:
```
ug           1
bo           1
my           2
dv           5
or           6
          ...
fr     1820095
es     2809891
und    2829073
tr     3695566
en     7650118
Length: 66, dtype: int32
```

[14]:
```
# plotting the tweet counts by language. This plot has only few of the major␣
↪languages
plot_no_of_tweets_by_language(tweet_count_by_language)
```



[37]:
```
# getting the percentage of tweets in the 6 top languages
np.
↪sum(tweet_count_by_language[tweet_count_by_language>=tweet_count_by_language['it']])/
↪np.sum(tweet_count_by_language)*100
```

[37]: 79.50911120725094

[41]:
```
# percentage of english tweets
tweet_count_by_language['en']/np.sum(tweet_count_by_language)*100
```

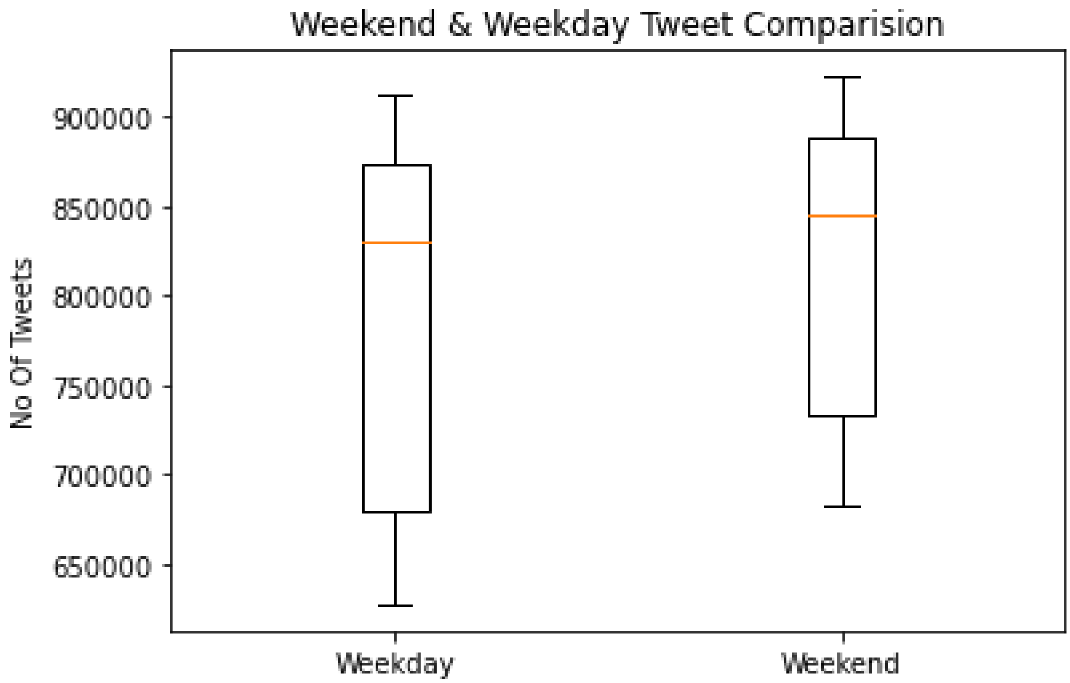
[41]: 30.85359115944112

2.0.6 Observation/Findings:

English is the most spoken language in Europe which is also evident from the tweet data. It can also be said that most tweets are from the English speaking countries of Europe that can be United Kingdom and Irelenad. English constitutes 30% of the total tweets made. It is followed by Turkish (15%), Und, Spanish, French, Italian, etc. These six languages constitutes 79.5% of the total tweets.

2.0.7 Comparing the number of tweets on a weekday to weekend using box plot

[43]:
```
weekday_data = df_tweets_by_day['count'][df_tweets_by_day['weekday']<=4]
weekend_data = df_tweets_by_day['count'][df_tweets_by_day['weekday']>4]
data = (weekday_data, weekend_data)
box_plot_weekend_to_weekday(data)
```



2.0.8 Observation

It can be said from the description of box plots that there is usually high twitter activity on weekends as compared to week days. It may be due to the fact that people could spare a lot of time for twitter owing to holidays on weekends. There is a larger spread of the number of tweets on a weekday,which means that tweet counts on weekday can vary a lot. It means that on some special occasions, weekdays can also have higher tweet activity.

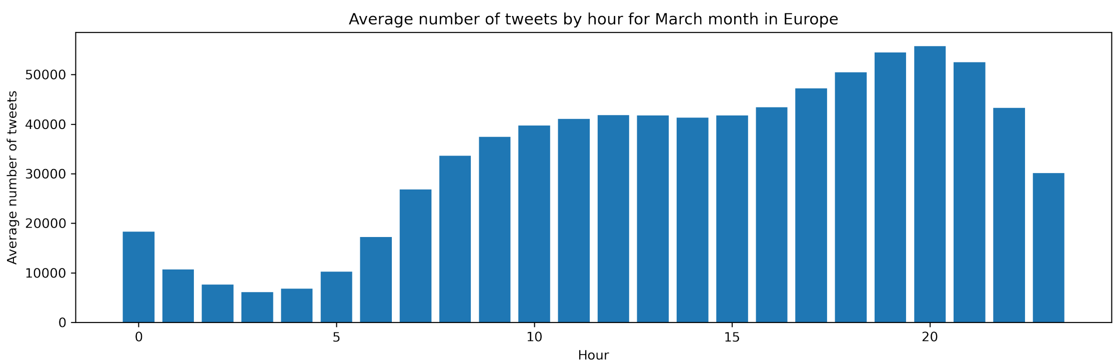
2.0.9 Plotting the average number of tweets at each hour of the day for March month

[16]:
```
avg_tweet_count_by_hour = get_tweet_count_by_hour()/31
avg_tweet_count_by_hour
```

[16]:
```
0     18338.451613
1     10715.903226
2      7640.161290
3      6105.838710
4      6823.129032
5     10237.967742
6     17228.032258
7     26801.967742
8     33642.387097
9     37439.451613
10    39739.032258
11    41089.935484
```

```
12    41830.709677
13    41753.290323
14    41326.258065
15    41785.483871
16    43431.064516
17    47264.064516
18    50450.258065
19    54481.096774
20    55760.548387
21    52532.774194
22    43307.903226
23    30109.870968
dtype: float64
```

[17]:
```
plot_tweet_count_by_hour(avg_tweet_count_by_hour)
```



2.0.10 Observation

Since the tweet data we have is with the respect to the time zone of UTC+0, we can do the analysis only with respect to this single timezone. However, there is only a time difference of 3 hours at the most within Europe. The twitter activity is very minimal during late night hours. It starts increasing in the morning and is quite significant during noon hours. However, it is highest during the evening hours from 6 PM to 10 PM. This would be true even with the max time difference Europe has.

3 Part 2. Mapping

1. Draw a map of Europe showing the location of the GPS-tagged tweets - these are tweets which have a "coordinates" field in the metadata. The exact form of the map is up to you, you can show individual tweets as points, use a heat map or a choropleth.
2. Explain any patterns you observe. using matplotlib or Folium (https://python-visualization.github.io/folium/) for integration with mapping platforms like open street maps.

[28]:
```
# Get the coordinates of tweets for 31 days
def get_tweet_coordinates():
    conn = sqlite3.connect('tweets2.sqlite')
    c = conn.cursor()
    all_tables = c.execute("SELECT name FROM sqlite_master  WHERE type='table'␣
 ↪and name like 'tweets%';")
    df_coordinates = pd.DataFrame([],columns=['lat', 'long'])
    for table in all_tables.fetchall():
        table_name = table[0]
        df = pd.read_sql_query("SELECT lat, long from {}".format(table_name),␣
 ↪conn)
        df.dropna(inplace=True)
        df_coordinates = df_coordinates.append(df, ignore_index=True)
    conn.close()
    return df_coordinates
```

We can easily get the coordinates of all the tweets on all the days of March using the function above. It gives geo locations of 1403355 tweets.

[17]:
```
df_coordinates = get_tweet_coordinates()
df_coordinates
```

[17]:
```
               lat       long
0        52.481388  13.435000
1        41.549028   2.103483
2        47.405188   8.129725
3        53.358689  10.208094
4        50.792653  -3.195868
…              …          …
1403350  42.784300  12.712800
1403351  54.000000  -2.000000
1403352  49.167517  -2.082838
1403353  51.530760  -0.080820
1403354  40.400000  -3.683330

[1403355 rows x 2 columns]
```

[35]:
```
#plotting the coordiantes on folium heatmap
plot_map(df_coordinates)
```

[35]:
```
<folium.folium.Map at 0x2657ce03af0>
```

3.0.1 Observation/Analysis

The places which are the hots spots of tweets can be identifed with the heatmap. The brightest yellow regions are the cities/countries which have larger popualation of tweeter users. These are mostly the megacities or capital cities of Europe. For eg., Barcelona, Oslo, Stockholm, Paris, Manchester, Liverpool, London, Munich, Berlin etc. It also highlights that UK had highest number of tweets in the month of March. It can be said that among all European countures, twitter is most popular in UK as it has highest number of tweets.

4 Part 3. Users

1. Make a histogram of tweets per user with number of users on the y-axis and number of tweets they make on the x-axis. Describe the distribution that you see.
2. Find the top-10 users by total number of tweets. Do you think any are automated accounts?
3. Plot the number of mentions per user and comment on it.
4. Study some of the users who are mentioned the most and try to understand why.

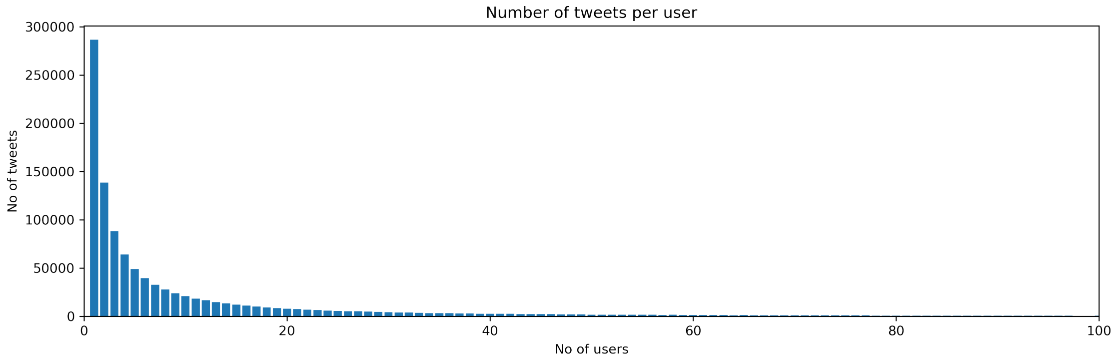
4.0.1 Histogram of tweets per user with number of users

[29]:
```
def get_no_of_tweets_per_user():
    # getting the no of tweets per user
    conn = sqlite3.connect('tweets2.sqlite')
    c = conn.cursor()
    all_tables = c.execute("SELECT name FROM sqlite_master  WHERE type='table'␣
↪and name like 'tweets%';")
    ser = pd.Series([], dtype=np.int64)
    for table in all_tables.fetchall():
        table_name = table[0]
        ser1 = (pd.read_sql_query("SELECT user_id from {}".format(table_name),␣
↪conn))
        ser1 = ser1['user_id'].value_counts()
        ser = ser.add(ser1, fill_value=0).astype(int)
    conn.close()
    return ser
```

[20]:
```
# getting the number of tweets per user
no_of_tweets_per_user = get_no_of_tweets_per_user()
no_of_tweets_per_user.sort_values()
```

[20]:
```
1236437364                 1
992745752                  1
992719134                  1
992718823                  1
992689470                  1
                       …
1384110594             11867
1232848641813024770    12012
928198040061710336     13623
550261599              21404
824637752574488576     37343
Length: 1103770, dtype: int32
```

[21]:
```python
# plotting the histogram
plot_no_of_tweets_per_user(no_of_tweets_per_user)
```



[ ]:
```python

```

4.0.2 Findings

The distribution in above plot is highly right-skewed i.e. it has positive skewness.It has a very long right tail. Mean would lie to the right of the median.

4.0.3 Top-10 users by total number of tweets.

[22]:
```python
# Top 10 users who have highest number of tweets
top = no_of_tweets_per_user.nlargest(10)
usernames = get_usernames(top.index.values)
top_tweeting_users = pd.DataFrame(top)
top_tweeting_users['user_name'] = usernames
top_tweeting_users = pd.DataFrame(top, columns=['no_of_tweets'])
top_tweeting_users['user_name'] = usernames
top_tweeting_users
```

[22]:

	no_of_tweets	user_name
824637752574488576	37343	Seen on OLIO
550261599	21404	infosrv
928198040061710336	13623	Animals Belize
1232848641813024770	12012	Koray Davulcu
1384110594	11867	L'hora catalana
160874621	8562	BB RADIO Playlist
161262801	8541	Radio TEDDY Playlist
824981479000305665	8516	Haykakan.top
1181921957522083840	7600	Mathieu Ronsard
253771137	7443	elena garcia santos

4.0.4 Explanation

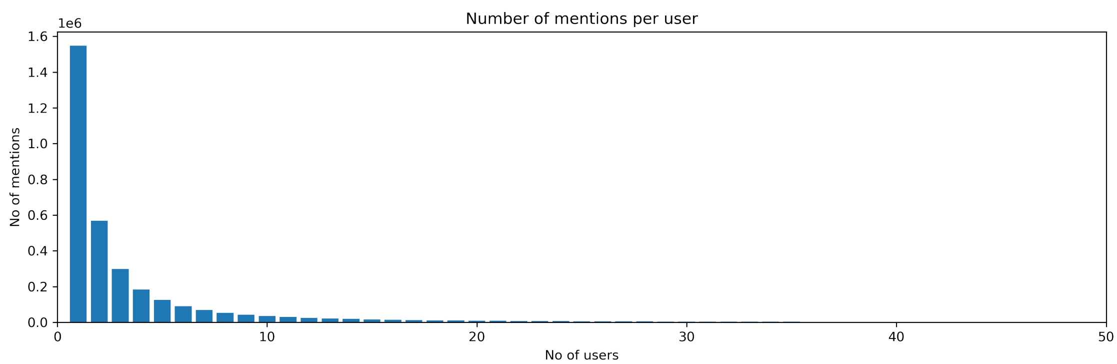
These top 10 users are surely some automated accounts. For eg, if we study the the top user ‘Seen on OLIO’, it can be clearly concluded that it is a food sharing charity app as it always tweets about the food or other products that can be shared so that there is no food waste. Similarly, second top user ‘infosrv’ always tweets about the IP adresses.Other top users are automated accounts of radio companies, wheather department, news channels, etc. These automated accounts are called bots.

4.0.5 Number of mentions per user

[23]:
```
# getting the number of mentions for a user
users_mentioned = get_no_of_mentions_for_a_user()
users_mentioned
```

[23]:
```
3131144855             41144
10228272               37243
216299334              36488
1016198241417822208    29979
335141638              27371
                       …
1186947922140827649        1
2576791                    1
1185323074260000770        1
1193273351516688386        1
147393579                  1
Length: 3352386, dtype: int64
```

[45]:
```
# plotting the number of mentions per user
plot_no_of_mentions_per_user(users_mentioned)
```



4.0.6 Comment

This plot is also right skewed and has a long right tail. There are only a few users who are mentioned the most.From the above plot, we can say they are not more than 20 and only 10 of them are the most popular ones. They must me some celebrities/politicians who drive most of the discussions on twitter and are tagged in tweets/retweets.

4.0.7 Most mentioned user

[25]:
```
# Top 10 users who are mentioned the most
top = users_mentioned.nlargest(10)
usernames = get_usernames(top.index.values)
top_mentioned_users = pd.DataFrame(top)
top_mentioned_users['user_name'] = usernames
top_mentioned_users = pd.DataFrame(top, columns=['no_of_mentions'])
top_mentioned_users['user_name'] = usernames
top_mentioned_users
```

[25]:

	no_of_mentions	user_name
3131144855	41144	Boris Johnson
10228272	37243	YouTube
216299334	36488	Piers Morgan
1016198241417822208	29979	Dr. Fahrettin Koca
335141638	27371	
25073877	23334	Donald J. Trump
68740712	19637	Pedro Sánchez
68034431	18119	Recep Tayyip Erdoğan
1201207366424752129	17988	AnimalDefenceBZ
270839361	13011	Matteo Salvini

Most mentioned users are the public figures like politicians, journalists, etc. Borris Johnson, Prime Minister of the UK, was the most mentioned user in the March. Apart from him, there are other politicians like Donald Trump (President of USA), Matteo Salvini (Former Deputy Prime Minister of Italy), Pedro Sánchez (Spanish Politician) who were mentioned the most. They are the ones who drive most of the discussions on twitter and are, hence, mentioned the most in the form of tweets or retweets.

5 Part 4. Events

1. Visually identify 3 days with unusually high activity in countries of your choosing. For example you could choose two days in the UK and one in France. Describe and justify how you locate tweets in countries and identify 'unusual days'.
2. Characterise each of these three days. Exactly how you do this is up to you, but for example you could: Display some indicative Tweets. Make a word cloud from the tweet text. Plot tweets locations on a map. Summarise the events you have detected and validate with some other source of data e.g. news articles.

To identify the days and countries with unusually high activity, first we could plot the day wise count of tweets in the country of our interest. For that I created a function called get_tweet_trends_for_country() as defined below. It gets the day wise count of a country. We can then plot this data (using function plot_tweets_vs_day_for_country()) and visually identify which day has highest number of tweets in that country in the month of March. The highest count of tweets could be an indicator of unusually hight activity. Then,word cloud of that particular day can be further analyzed to characterise the day.

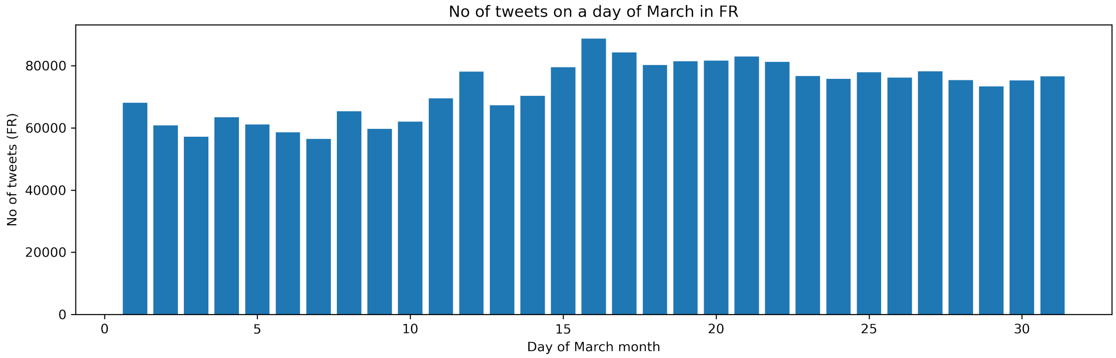
[10]:
```
def get_tweet_count_by(country, day):
    """Return the tweet count on a day in a country"""
    conn = sqlite3.connect('tweets2.sqlite')
    c = conn.cursor()
    table_name = 'tweets_2020'+day
    df = pd.read_sql_query("SELECT bound_country from {}".format(table_name),␣
 ↪conn)
    count = np.sum(df['bound_country']==country, axis=0)
    conn.close()
    return count

def get_tweet_trends_for_country(country):
    """Return the day wise count of a country"""
    counts = [get_tweet_count_by(country,'030'+str(i)) if i<10 else␣
 ↪get_tweet_count_by(country,'03'+str(i))
              for i in range(1,32)]
    return counts

def plot_tweets_vs_day_for_country(country, y):
    """Plot the tweet count with a day of March in a country"""
    fig, axes = plt.subplots(nrows=1, ncols=1, figsize=(12,4), dpi =300)
    axes.bar(range(1,len(y)+1), y)
    axes.set_xlabel('Day of March month')
    axes.set_ylabel('No of tweets ({})'.format(country))
    axes.set_title('No of tweets on a day of March in {}'.format(country))
    fig.tight_layout()
    plt.show()
```

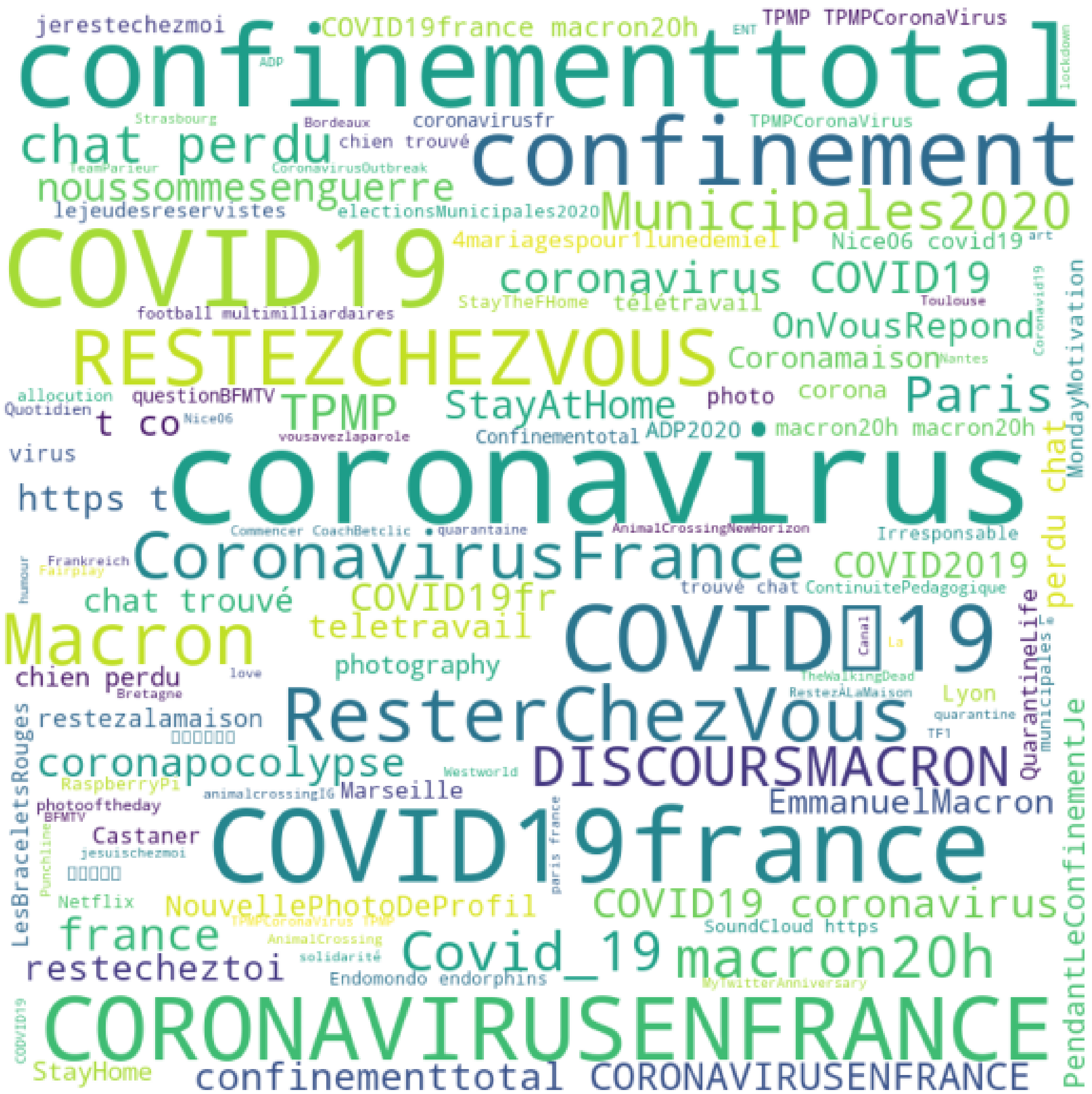
For eg., I was interested to check for France. I get the below plot, which depicts that there is some high activity on 16th and 17th of March. There is usually high tweet activity on weekends as we have observed before. However, these were weekdays. Getting the most trended words or hashtag words could give an idea of the cause of such high activity on twitter. Word cloud can be used for this purpose.

[11]:
```
country = 'FR'  # 16 and 17th despite being weekdays
counts = get_tweet_trends_for_country(country=country)
plot_tweets_vs_day_for_country(country, y=counts)
```



Below is a word cloud of hashtag words on 16th and 17th of March in France. Overall, March was the month when onset of COVID-19 pandemic was at the peak globally. So, COVID related trends were the most common in this month. In France also, tweets related to 'restezchezvous', 'COVID19france', 'Quarantine', 'confinement' were common on 16th and 17th of March. Apart from this, there were also some other specefic trendy topics. 16th of March can be characterised as day of Municipal elections in France because of hashtag words like 'Municipales2020'. It turns out to be true when verified in news as 15th to 28th June was a period of municipal election in France.

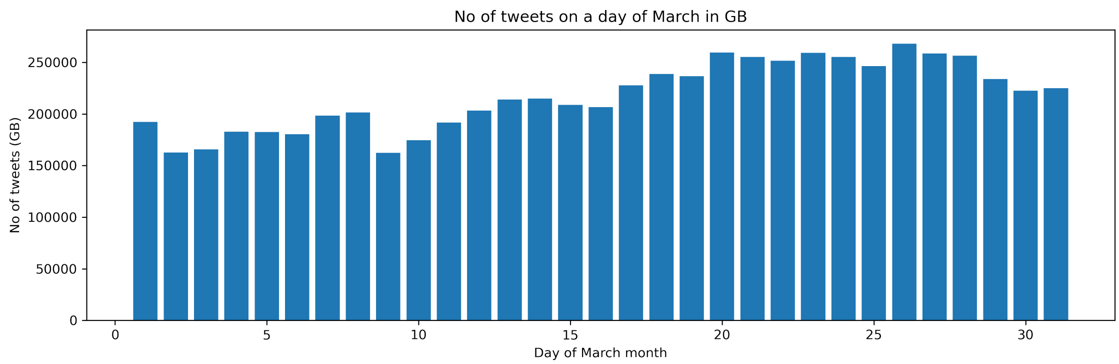
[12]:
```
trend_words = get_tweet_trends(country='FR', day='0316')
plot_word_cloud(trend_words)
```



The wordcloud for 17th March in France is shown below. On a similar note, 17th March can be characterized as Pekin Express day as it was trending on that day apart from COVID-19. Pekin Express, also known as Beijing express is a reality TV show, which has its 15th anniversary on this day. A special contest edition was telecasted on this day.

[13]:
```
trend_words = get_tweet_trends(country='FR', day='0317')
plot_word_cloud(trend_words)
```

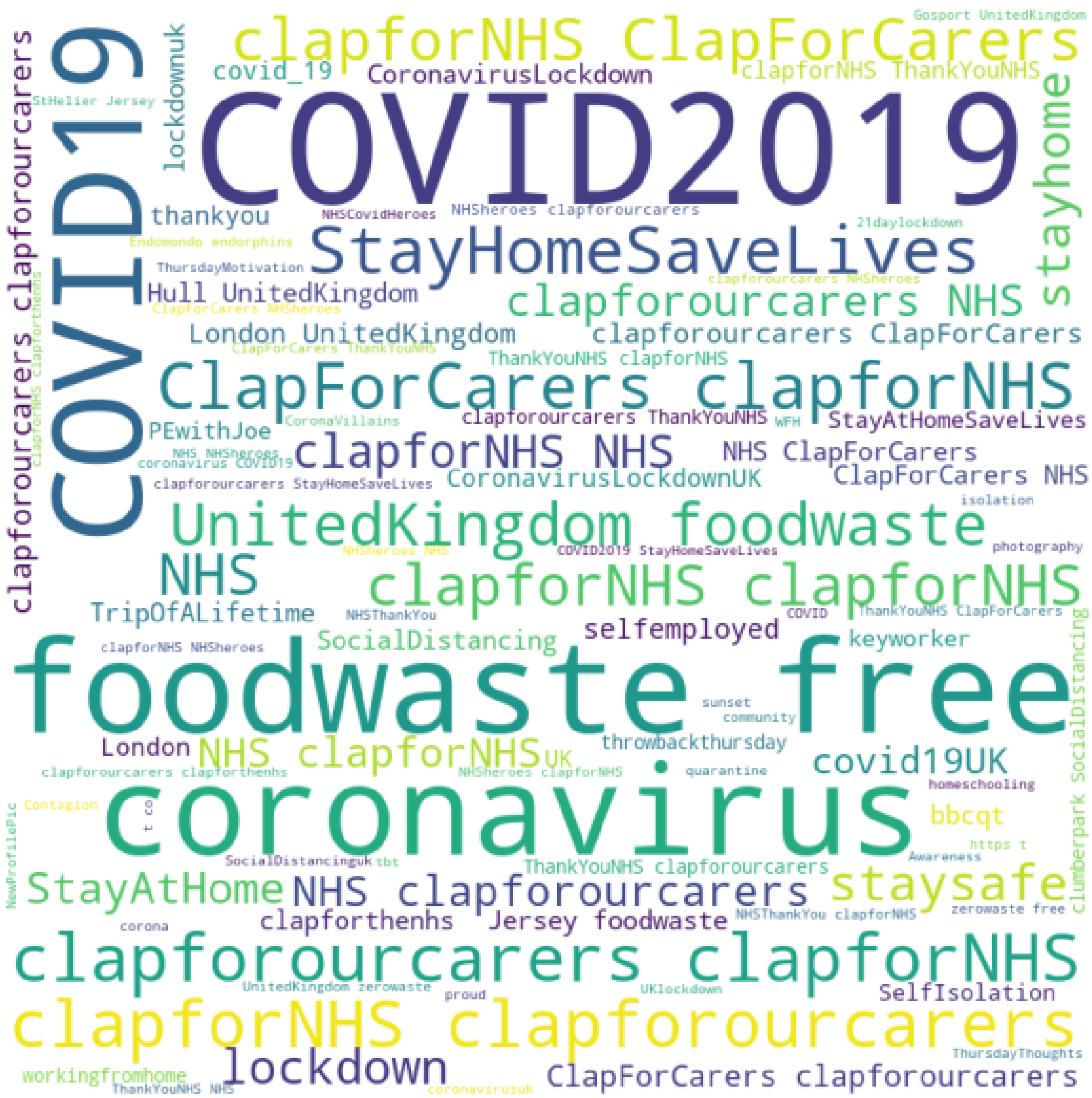

RESTEZCHEZVOUS
attestationsurlhonneur
COVID19
photo
ResterChezVous
StPatricksDay
coronavirus COVID19
CeSoirChezBaba
coronavirusfr
confinement
merci
battlephoto
corona
animalcrossingIG
AlloBaba
CORONAVIRUSENFRANCE
minister
chat trouvé
VirginTonic
beardporn beard
Euro2020
confinementtotal
coronavirus
Macron
gouvernement
Parisien
télétravail
4mariagespour1lunedemiel
perdu chat
COVID19fr
CoronavirusFrance
beard muscle
musclecouple
SolidariteCOVID19
coronavirus Covid_19
COVID19
paris
CONFINEMENTJOUR1
teletravail
Municipales2020
Quotidien
Buzyn
guerre
restezalamaison
Covid_19
QuarantineLife
COVID19 coronavirus
jerestechezmoi
OnVousRepond
NouvellePhotoDeProfil
attestation
COVID19france
restecheztoi
OnApplaudit
ConfinementGeneral
RaspberryPi
solidarité
Bordeaux
TPMP CeSoirChezBaba
NousSommesEnGuerre
Np
COVID2019
PekinExpress
art
PekinExpress PekinExpress
Covid_19fr
stayhomechallenge
Toulouse
PekinExpress M6
jour1
coronavirus
Marseille
chien perdu
Confinementotal
photography
StayAtHome
TPMP
France
TPMPCoronaVirus
chat perdu
t co
stayhome
https t
Strasbourg



The word cloud of 26th March has hasthtag words like 'Clap for Carers' apart from COVID. This day was actually celebrated as a day of tribute to the NHS workers, who were sacrificing so much for the general public due to COVID emergency.

Note: FoodwasteFree hashtag trends can be verified to be generated by some automated accounts, hence it was ignored.

[15]:
```
trend_words = get_tweet_trends(country='GB', day='0326')
plot_word_cloud(trend_words)
```



6 Part 5. Reflection

Critically reflecting on using Twitter data to find events. Discussion on below points: 1. The strengths and weaknesses of Twitter as a data source. 2. Biases in the data 3. Ethical and legal concerns

Social media has interesting use cases in a variety of research disciplines, such as media, communication, sociology, political science, computer science, engineering, etc. Twitter being one of the most used social media platforms remains an appropriate choice to get data to conduct such research [1]. Twitter data can be helpful in analyzing and management of emergency situations or extreme circumstances. These circumstances can be riots, natural disasters, and other crisis events. Authorities concerned with emergency management can utilize Twitter data to improve their position in responding to emergency situations. Twitter data contain valuable metadata, including geospatial data, such as precise latitude and longitude coordinates of the tweet location. Tweeter API is an open and powerful API, which makes it easy for researchers and developers to use it. Twitter has a strong hashtag culture that makes it easy to center the research on a particular topic.

However, there are also some weaknesses of Twitter data. It is not always true that users will discuss a trendy or interesting topic on Twitter. So, it is not always possible to gauge an event that is in trend from the tweets. Using hashtags or certain keywords to retrieve data related to a particular topic may not always work. Language also poses a challenge when you want to retrieve the data in your topic of interest across multiple geographical locations. More problematic could be the bias in Twitter data. For e.g., filtering the tweet data by certain keywords or hashtags may introduce a bias. Also, link-baiting in popular hashtags amounts to spam at a large scale. It is difficult to identify a real or fictitious user. Fictitious or automatic accounts are set up to increase users' followers and popularity. So, there are a lot of fake accounts and retweets on Twitter[3].This was evident from the word clouds studied in this assignment. For e.g. FoodwasteFree hashtag trends can be verified to be generated by some automated accounts.

There also ethical and legal concerns associated with the Twitter data. It is easy to get the Twitter data through API, but it cannot be said for sure that how many users moderated their behavior accordingly while posting on Twitter as a public space. Therefore, whether Twitter, as an online space, is public or private is a topic of debate. When collecting and using the users' data, it may not be possible to obtain consent from the users because of the volume of the data. Similarly, it becomes difficult to reproduce tweets in the research publications without having consent from the users.[2]

In conclusion, Twitter as a data source has many benefits, but it also poses many ethical and legal challenges. These challenges could be related to the users' consent to use their data. There can also be problems of spamming and bias owing to fake and automated accounts. Too much reliance on hashtag words may also be a cause of bias.

References:

[1] Ngai, E.W., Tao, S.S. and Moon, K.K., 2015. Social media research: Theories, constructs, and conceptual frameworks. International journal of information management, 35(1), pp.33-44. https://doi.org/10.1016/j.ijinfomgt.2014.09.004

[2] Moeen Uddin, M., Imran, M. and Sajjad, H., 2014. Understanding Types of Users on Twitter. arXiv, pp.arXiv-1406.

[3] Castillo, C., Mendoza, M. and Poblete, B., 2011, March. Information credibility on twitter. In Proceedings of the 20th international conference on World wide web (pp. 675-684). https://www.researchgate.net/publication/221023878_Information_credibility_on_Twitter