Twitter Data Analysis – Introduction to Data Science

30 November 2022

Contents

Introducing Twitter Data Set

Exploring and analysing a large dataset of tweets collected from the Twitter API. The time duration of the tweets analysed is in the period June 1st to June 30th, 2022. The data contains all the tweets with geographical coordinates in the Europe region with the (Longitude, Latitude) combinations of the lower-left corner at (-24.5, 34.8) and the upper-right at (69.1, 81.9).

Modules and Paths

A working directory has been changed from the default in the first step to carry out the analysis in the single and same folder which contains the data set. All the necessary modules are imported which are used in the entire analysis.

```
import os
os.chdir("C:\\Users\\TEJAS KUMAR V URS\\Desktop\\MSc Data Science\\Introduction to Data Science\\Course work\\TwitterJune202

import pandas as pd
import numpy as np
import matplotlib as mpl
from matplotlib import pyplot as plt
import matplotlib.gridspec as gs
import plotly.express as px
import geopandas as gpd
import mpl_toolkits

import zipfile
import json
import country_converter as coco
from wordcloud import WordCloud, STOPWORDS

path1 = "C:\\Users\\TEJAS KUMAR V URS\\Desktop\\MSc Data Science\\Introduction to Data Science\\Course work\\TwitterJune2022"
path2 = os.listdir(path1)
```

Fig 1: Modules & Paths

Part 1: Basis Stats

Task 1: Number of Tweets

The total number of tweets in the data set is **150,40,387** with 6,839 duplicates present in the dataset, the unique number of tweets is 150,33,548. There are few anomalies in the dataset too,

- The number of lines in the data set is 150,40,709 but most of them are null values and do not have any tweets associated with them which does not count as a tweet.
- Time stamp in the data set entity is 1 hour behind, that is due to the time difference between UTC and GMT standards, the analysis is carried out basis GMT standards.

The entire analysis will be carried out without removing the duplicate tweets from the dataset because of its quantum which is very negligible around 0.045%. Performing data deletion operations in the main dataset with just this bare minimum of duplicates might pose a risk.

Code logic and snippets:

The zipped files which contained all the JSON files were extracted using the *Zipfile* module and all JSON files were put in one single folder for analysis. Since the data is huge, creating one big *DataFrame* is not a cost-effective method. The entire analysis is carried out based on readlines logic which basically means we take specific data from each of the tweets severally and store it in data structures. For this task, counting the number of tweets, a *Set* is used for storing unique tweets and a *List* is used for storing all the tweets by counting the unique ID of tweets in the dataset "id_str"

```
tweets_set = []
for i in range(0,len(JSON_files)):
    temp = path1 + '\\' + path2[i]
    with open (temp,'r') as f:
        for lines in f:
            tweets = json.loads(lines)
            try:
                tweets_set.append(tweets['id_str'])
            except KeyError:
                pass
    tweets_count = len(tweets_set)
```

Fig 2: Number of Tweets

Task 2: Time - Series plot for day-wise Tweets

There is no trend that can be seen in the Time–Series plot for day-wise tweets, but the number of tweets at the start of the month is at the peak and post that there are a lot of fluctuations and does not have any kind of direction. The 3 lowest activities were observed on the 13th, 20th and 27th of June.

Code logic and snippets:

The number of tweets was calculated by the same logic as readlines and was stored in a *List.* The list was later dumped inside a NumPy array, and the array was divided into 30 individual arrays since each file represents hourly tweets inside the dataset. These individual arrays were summed and plotted using Matplotlib.

```
tweets = np.array(count_twitterlines)
tweets_DW = np.array_split(tweets, 30)

list_daywise_tweets = []
for i in range(0,len(tweets_DW)):
    list_daywise_tweets.append(np.sum(tweets_DW[i]))

index = np.arange(1,31)

fig, ax = plt.subplots(figsize= (10,5))
ax.set_title("Number of Tweets per day", fontsize = 15, fontstyle='italic')
ax.plot(index,list_daywise_tweets)
ax.scatter(index,list_daywise_tweets)
ax.set_xlabel('Days', fontsize=12)
ax.set_ylabel('No. of Tweets', fontsize=12)
ax.set_xlim(0,31)
ax.set_xticks(np.arange(0,31)
```

Fig 3: Time-Series code

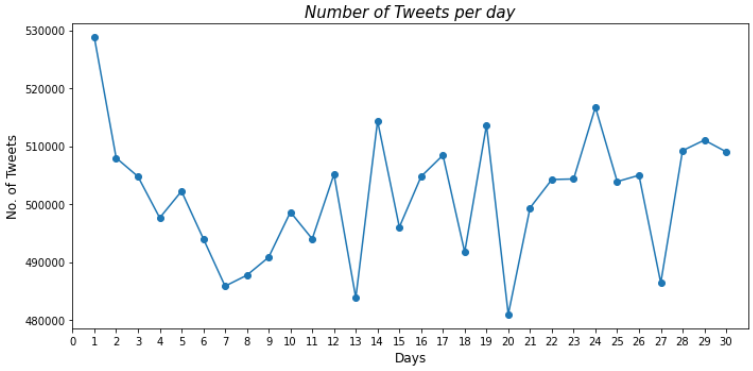


Fig 4: Number of Tweets per day

Task 3: Box-Whisker plot for Weekends and Weekdays tweets

Through the obtained Box-Whisker plot we can clearly see that the median of the number of tweets on both weekdays and weekends is almost the same at an approximate value of 5,45,000 but, the range of the number of tweets varies a lot. The quartiles in both weekend and weekday data have a certain ratio to it owing to the number of days in both. In short, the quartiles in weekend and weekday tweets are proportional to each other with a ratio.

Code logic and snippets:

The data already obtained which had day-wise tweets count was put in a *Pandas DataFrame* and with the help of *Pandas Time-Series* a new column was created that labelled days in the month of June as 'Weekends/Weekdays.' To perform this task a special function "***.strftime('%w')***" [1] was used which converts days into the day numbers in a week, for example, Sunday is considered the first day of the week and numbered as 0. Using a simple *for* loop these week numbers can be categorized as Weekends/Weekdays. Two different functions are called simultaneously to dump the weekend and weekday data in separate *Lists.*

```
days = pd.date_range('2022-06-01','2022-06-30', freq='D').strftime('%w')
df['day number'] = days

wd=[]
for i in range(0,len(df)):
    if int(df['day number'][i]) == 0:
        wd.append('Weekend')
    elif int(df['day number'][i]) == 6:
        wd.append('Weekend')
    else:
        wd.append('Weekday')
df['WD'] = wd

def weekday_data():
    df1=[]
    for i in range(0,len(df)):
        if df['WD'][i] == 'Weekday':
            df1.append(df['tweets'][i])
    return df1

def weekend_data():
    df2=[]
    for i in range(0,len(df)):
        if df['WD'][i] == 'Weekend':
            df2.append(df['tweets'][i])
    return df2
wd_data = weekday_data()
we_data = weekend_data()


fig, ax = plt.subplots(figsize=(6,8))
ax.set_title('Box-Whisker plot',fontsize = 15, fontstyle='italic')
ax.boxplot(wd_data, vert=True, positions=[0], labels=["Weekday_data"], widths=0.3)
ax.boxplot(we_data, vert=True,positions=[1], labels=["Weekend_data"], widths=0.3)
```

Fig 5: Weekday-Weekend tweets

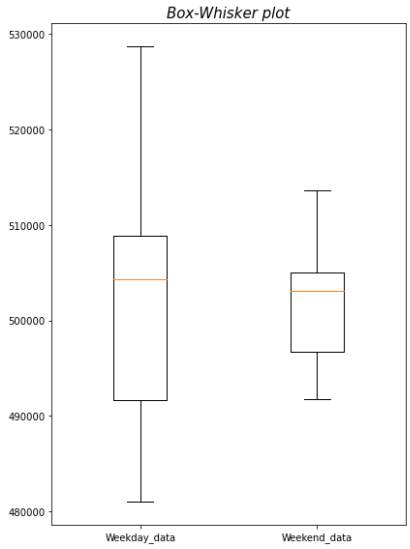


Fig 6: Weekday-Weekend Box Whisker plot

Task 4: Number of tweets by hour averaged over weekdays

- The number of tweets increases gradually and attains a peak during the evening between 6 PM - 9 PM.
- Post 9 PM number of tweets drops significantly and continues to dip till 3 AM and gains momentum post 3 AM
- The total number of tweets on weekdays and the average number of tweets on Weekdays, when viewed hourly wise, follow the same trend.

Code logic and snippets:

Following the same steps in the previous task and adding one more column to the DataFrame for an hour. *Pandas groupby* function is used to group tweet counts by hours and average them to plot our required time-series plot.

```
df1 = df.groupby(["WD","Hours"]).sum()
df2 = df.groupby(["WD","Hours"]).count()
df3 = df1/df2

# Fetching only Weekday data
df = df3.loc['Weekday']

index = np.arange(0,24)
fig, ax = plt.subplots(figsize=(10,5))
ax.set_title("Number of Tweets per Hour averaged over Weekdays", fontsize = 15, fontstyle='italic')
ax.plot(index, df["tweets"])
ax.scatter(index, df["tweets"])
ax.set_xlim(0,24,1)
ax.set_xticks(index)
ax.set_xlabel('Hours', fontsize=12)
ax.set_ylabel('Average No. of Tweets', fontsize=12)
```

Fig 7: Number of tweets per hour – weekdays code

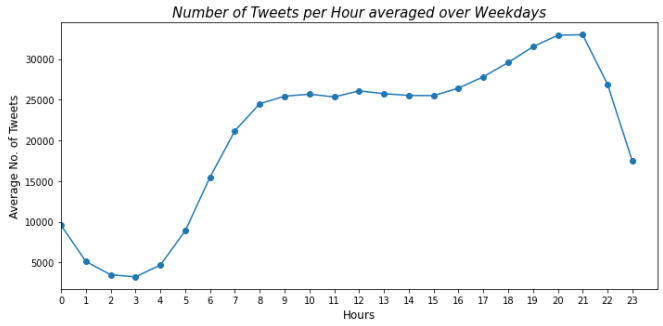


Fig 8: Number of tweets per hour – weekdays

Part 2: Users

Task 1: Number of users versus Number of tweets

The number of tweets ranged from 1 and 13,382 in a month's period by an individual. Since there are few outliers who have tweeted above 10,000 per month and most of the population behaving pragmatically with respect to the number of tweets, the graph plotted will have a very lean and sharp line in the starting and goes invisible as the tweets count increases and will be difficult to view it without an alternative version. An alternative "Y-axis trimmed version" is plotted which trims the certain user's count in the y-axis and gives kind of a zoomed view for better visibility. With this second version, we can clearly conclude that it is a *Right-skewed* graph.

Code logic and snippets:

In the Twitter dataset, under the user entities 'screen_name' of the users which are unique to individual users were fetched by line-by-line approach and stored in a *Dictionary.* By importing *Counter* from the *collections* library, the number of unique users were counted and plotted against the number of tweets by using *hist* plot from the *matplotlib* library to get our desired plot.

```
user = {}
# using for loop and json.loads,the following code was used to append user names
# used exception handling to remove KeyErrors where tweets are null
 user.append(tweets['user']['screen_name'])

# for calculating number of tweets by individual users
from collections import Counter

user_count = Counter(user)
u = []
for k,v in user_count.items():
    u.append(v)

# Main plot
ax[0].hist(u, bins=np.arange(0,13500,100))
ax[0].axvline(max(u), color='orange', label = 'Maximum number of Tweets by a user = {:}'.format(max(u)),
              linewidth=1, linestyle=':')
ax[0].axvline(min(u), color='red', label = 'Minimum number of Tweets by a user = {:}'.format(min(u)),
              linewidth=1,linestyle=':')

# Y-axis trimmed version plot
ax[1].hist(u, bins=np.arange(0,13500,100))
ax[1].axvline(max(u), color='orange', label = 'Maximum number of Tweets by a user = {:}'.format(max(u)),
              linewidth=1, linestyle=':')
ax[1].axvline(min(u), color='red', label = 'Minimum number of Tweets by a user = {:}'.format(min(u)),
              linewidth=1,linestyle=':')
```

Fig 9: Number of users vs tweets – code

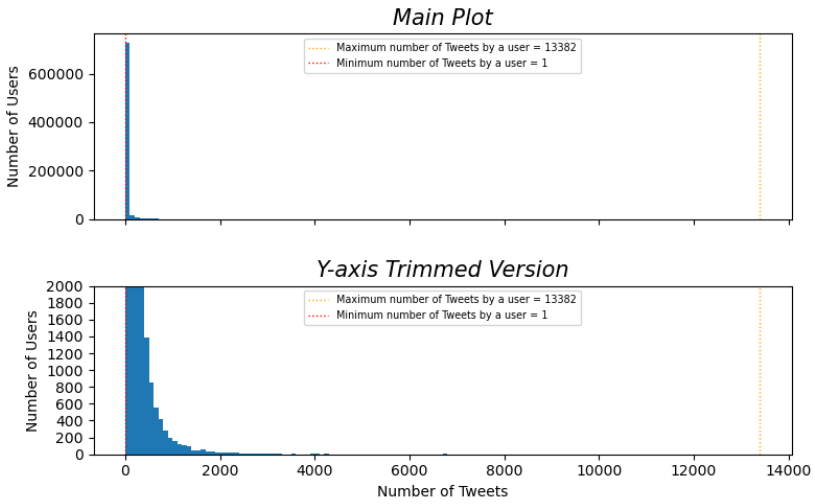


Fig 10: Number of users vs tweets

Task 2: Top-5 users by number of tweets

The top-5 users by their number of tweets in the month of June 2022 are:

1. Kardeimcin1 – 13,382

Doing a simple math for first user; 13,382 tweets per month is around 18.5 tweets per hour that basically mean he tweets every 3.3 minutes which looks kind of non-viable. This math and the fact that it is not a verified account are sufficient to consider this profile as a bot. When read between the lines through his tweets it contained information related to prisons, criminal cases and showing his voice against Law in Turkey. With this we can also say it can be a bot account created for accelerating the reach in twitter world.

2. DailyNews79 – 12,533

In the second account, since it is a profile related to news and usually media personnel in social media tweet every single news across the world, there is a slight possibility that it is handled by bunch of employees in the organization and these people tweet according to the news reports they get. Otherwise, this can also be a bot which is automated with all the news articles generated in the web.

3. c_antolic – 11,632

Cannot conclude is it is a bot or human as it contains some random words and single word tweets in most of its tweets. Concluding as human who is retarded and tweeting relentlessly since there is no agenda attached to it.

4. HoraCatalana – 11,305

It is a bot which just tweets times in Spanish time zone and has total 1.3 Mn tweets lifetime and being a 9 years and 7 months old account tweets around 16 tweets per day.

5. Minijobanzeigen – 10,087

In the profile, info itself says it is an automated profile. So, it is very clear that it is a bot.

Code logic and snippets

Users dictionary from previous task was converted to *Pandas DataFrame,* using *Groupby* function, number of tweets by users were calculated and by sorting, top-5 users were identified.

```
dfu = pd.DataFrame(user, columns=['user_name'])
dfu ['count'] = 1
dfc = dfu.groupby(['user_name']).count()
dfc.sort_values(by=['count'],ascending=False)
```

Fig 11: Top-5 Users number of tweets

Task 3: Top-5 users with the most mentions

1. YouTube – 20,700
2. RTErdogan – 17,973
3. BorisJohnson – 16,117
4. elonmusk – 10,570
5. GBNEWS – 7,902

Code logic and snippets

Under the entities section in the dataset 'user_mentions' has all the profile mentions by a user. Since there are tweets which does not have 'user_mentions' entity, an intermediate data storage was created and if logic was used to fetch 'screen_name' under 'user_mentions' entity wherever it contained. By using similar approach like previous task, top-5 users who have highest mentions were identified.

```
# using for loop and json.loads,the following code was used to append user names
# used exception handling to remove KeyErrors where tweets are null
temp_users = tweets['entities']['user_mentions']
if temp_users:
    for i in temp_users:
        mentioned_users.append(i['screen_name'])
    else:
        pass

# Pandas dataframe for calculation
dfm = pd.DataFrame(mentioned_users, columns=['mentioned_usernames'])
dfm ['count'] = 1
dfmc = dfm.groupby(['mentioned_usernames']).count()
dfmc.sort_values(by=['count'],ascending=False)
```

Fig 12: Top-5 users mentioned in tweets – code

Task 4: 4 Countries mentioning each other

Mentions	Count	Mentions	Count
UK_mentions_UK	3,10,524	Turkey_mentions_UK	1,888
UK_mentions_Turkey	1,715	Turkey_mentions_Turkey	1,76,585
UK_mentions_France	6,497	Turkey_mentions_France	1,105
UK_mentions_Spain	8,596	Turkey_mentions_Spain	1,285
Spain_mentions_UK	8,489	France_mentions_UK	5,797
Spain_mentions_Turkey	338	France_mentions_Turkey	683
Spain_mentions_France	4,018	France_mentions_France	1,08,071
Spain_mentions_Spain	1,67,614	France_mentions_Spain	2,707

- A country mentioning themselves are always high in number compared to them mentioning another country.
- A country other than mentioning themselves, the highest number of times they mentioned was related to UK.
- Number of mentions received by Turkey has lowest than other countries.

Code logic and snippets:

By country_mentions_country, it is considered that someone from one country mentions a person from other/same country, the analysis is carried out basis this logic.

At first, all the screen names and their country codes were captured inside a dictionary, this can be fetched from *user* entity for storing unique username and *place* entity for storing that tweets'/users' association with the country. Here *place* entity was used for fetching country code because it is the location object which is associated with the tweet according to Twitter API document, though there is coordinates entity which gives exact tweet location, computation will be difficult. Tweeted user along with his mentioned user in the tweet were stored in different dictionary, by storing these two different dictionaries in separate *DataFrame* and using *pd.merge* we can calculate the country codes of all the users and mentioned users in 2nd dataframe using 1st dataframe, all the exception cases were handled during the fetching stage like KeyError(for null values inside the tweet object). Once all the data was dumped into a single dataframe which had user, mentioned_users and their country codes, *df.groupby* was used to understand the combinations and their counts. Though this will not give us mentioned users' country code if they have not tweeted in June month of 2022, this was the optimum and best possible way to get to the solution.

```
# For fetching country codes of all users who tweeted
user_cc = {'screen_name': [], 'country': []}
user_cc['screen_name'].append(tweets['user']['screen_name'])
if tweets['place']:
    user_cc['country'].append(tweets['place']['country_code'])
else:
    user_cc['country'].append('Null')

# For fetching User and Mentioned User
user_dict = {'main_user': [], 'mentioned_user': []}
temp_users = tweets['entities']['user_mentions']
for i in temp_users:
    if temp_users:
        user_dict['mentioned_user'].append(i['screen_name'])
        user_dict['main_user'].append(tweets['user']['screen_name'])
    else:
        pass
```

Fig 13: Mentioned Users - code

```
# 1st dataframe of users and their country codes
ds = pd.DataFrame(data = user_cc)
ds1 = ds.drop_duplicates().reset_index(drop=True)
ds2 = ds1.drop_duplicates(subset=['screen_name']).reset_index(drop=True)

# 2nd dataframe with users and their mentioned codes
# along with dataframe computation for finding combinations
dum = pd.DataFrame(data = user_dict)
dq1 = dum.merge(ds2, how='left', left_on='main_user', right_on='screen_name')
dq2 = dq1.merge(ds2, how='left', left_on='mentioned_user', right_on='screen_name')
dq2['combinations'] = dq2["main_user_cc"].astype(str) + dq2["mentioned_user_cc"].astype(str)
occur = dq2.groupby(['combinations']).size()
```

Fig 14: DataFrame operation for combinations

Part 3: Mapping

Task 1: Europe map that displays tweets coordinates of dataset

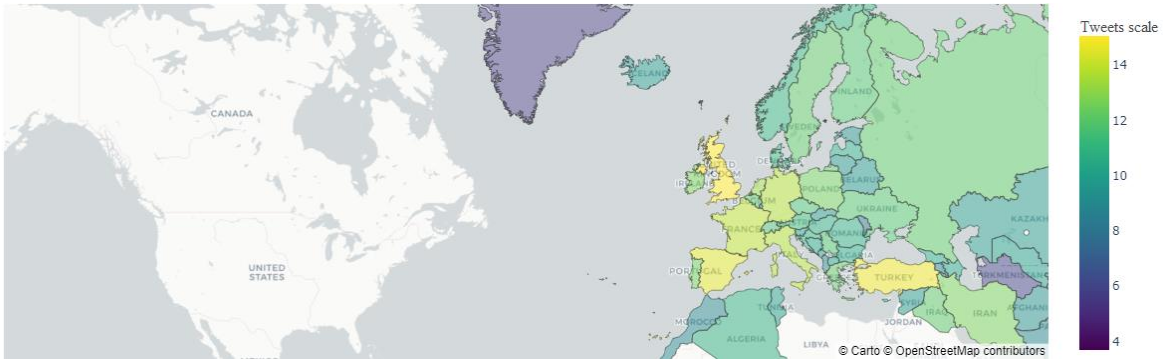


Fig 15: Europe map with number of tweets

Code logic and snippets

All the coordinates as per the requirement were fetched from the *coordinates* entity in dataset and was stored in DataFrame with Latitude and Longitude on different columns. Using *GeoJson* file for mapping and *Nominatim* function from *geopy.geocoders* to calculate country codes the above displayed map was plotted with the help of *choropleth* map in *plotly.express* module.

```
# for finding country codes from Lat-Long combinations
from geopy.geocoders import Nominatim
coord = f"{row['Latitude']}, {row['Longitude']}"
sleep(1)
location = geolocator.reverse(coord, exactly_one=True, language='en')
if not location:
    print('Failed with coord: ', coord)
    row['city'], row['state'], row['country2'] = None, None, None
    return row
address = location.raw['address']
city = address.get('city', '')
state = address.get('state', '')
country = address.get('country', '')
row['city'] = city
row['state'] = state
row['country2'] = country
return row
```

Fig 16: Country code calculator

Task 2: Patterns observed on the Europe tweets map

- Highest tweets from a region: United Kingdom > Turkey > Spain > France
- Though Russia is largest country in area, tweets coming from this region is moderate

Task 3: CDF of the Bounding box diagonals

Euclidean distance was used to calculate the distance within bounding boxes coordinates.

- The minimum distance from which the tweets start is at 0.75.
- As the distance between bounding box diagonals increases, number of tweets also increases.

Code logic and snippets:

All the coordinates in the bounding box attribute were fetched and stored inside a list and converted to *numpy array* later for calculating Euclidean distances. Euclidean distance was calculated using a simple function called *np.linalg.norm* from numpy library, which calculates distances between two diagonal points of all the bounding boxes present in the tweets. A simple cumulative histogram was plotted using *matplotlib* library.

```
#Converting  list to numpy array and Calculating Euclidean distances
bb_diagonals = np.array(bb_coords)
euclidean_distances = []
for i in range(len(bb_diagonals)):
    euclidean_distances.append(np.linalg.norm([bb_diagonals[i][0][0]-bb_diagonals[i][0][2]]))
```

Fig 17: Bounding Box - CDF code

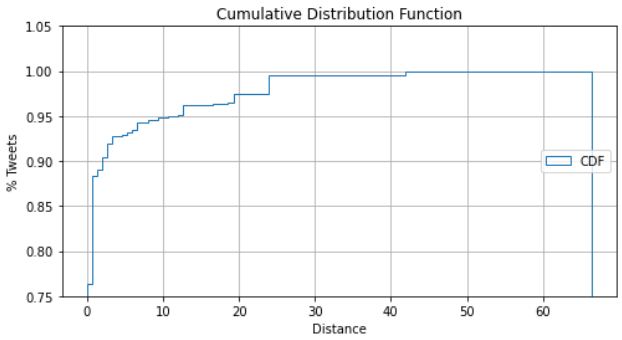
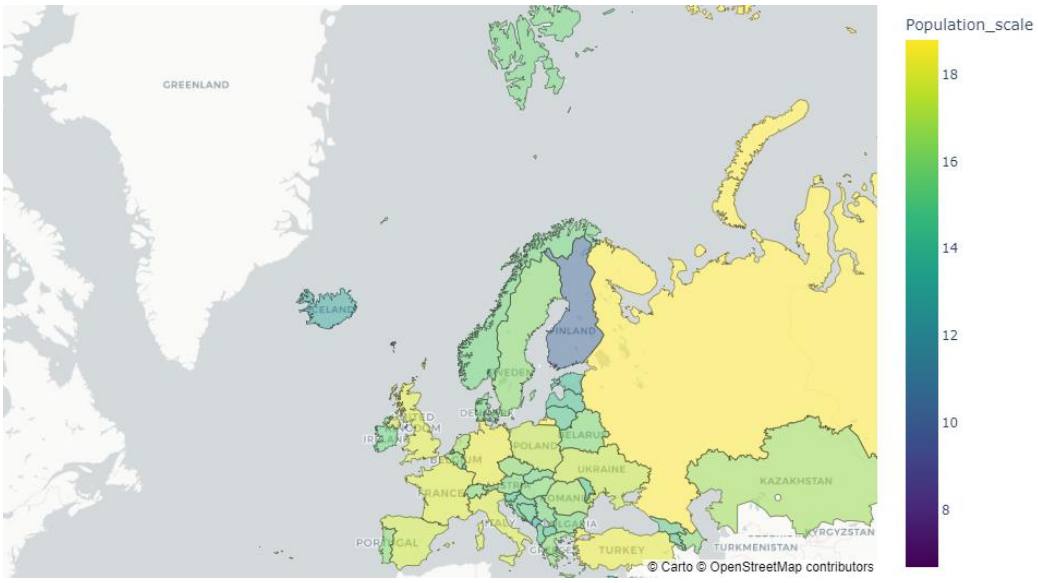


Fig 18: CDF - Bounding Box Diagonals

Task 4: Additional Spatial dataset for comparison

For additional spatial dataset, country wise population map was plotted for comparison with our Twitter data. The population data was collected from Wikipedia which has data according to 2021 census done by UN [2]. Few of the points that were observed after the comparison are as follows

- Though Russia has highest population, number of tweets coming from this region was lesser compared to other regions.
- United Kingdom on the other hand with medium scale on population, has highest number of tweets.
- Countries like Turkey, Portugal and France, Iceland have proportional rate of tweets with respect to their population.
- Finland at the bottom scale in population is at the top scale in number of tweets.



Code logic and snippets:

Code was same as that of 3.1 with just Population data considered instead number of tweets.

Part 4: Events

Task 1: 3 days with unusually high activity

For unusually high activity in a particular country and date was found by checking ***#hashtag*** trends. United Kingdom, Turkey and Italy were considered for this analysis and below mentioned unusual activities were found.

Date	Country	#hashtags trended	Count of tweets
4th June 2022	The United Kingdom	#PlatinumPartyatthePalace	1675
28th June 2022	Turkey	#CüneytArkın	2121
22nd June 2022	Italy	#ThePulpitRestoration	623

Code logic and snippets:

All the hashtags from *entities* entity were fetched by country and day wise and were stored inside a *python dictionary* and later dumped inside a *DataFrame.* By using a simple *groupby* in pandas library number of hashtag tweets with dates were computed and sorted by descending to get the highest activity.

```
# Creating dictionary with all the hashtags and their date of trends
UK_hashtags = {'tags':[],'date':[]}
Turkey_hashtags = {'tags':[],'date':[]}
Italy_hashtags = {'tags':[],'date':[]}

if tweets['entities']['hashtags']:
    temp_hashtag = (tweets['entities']['hashtags'][0]['text'])
    if tweets['place']['country_code'] == 'GB':
        UK_hashtags['tags'].append(temp_hashtag)
        UK_hashtags['date'].append(tweets['created_at'])
    elif tweets['place']['country_code'] == 'TR':
        Turkey_hashtags['tags'].append(temp_hashtag)
        Turkey_hashtags['date'].append(tweets['created_at'])
    elif tweets['place']['country_code'] == 'IT':
        Italy_hashtags['tags'].append(temp_hashtag)
        Italy_hashtags['date'].append(tweets['created_at'])

# Pandas DataFrame code for finding high activity
duk = pd.DataFrame(data=UK_hashtags)
duk['day'] = pd.to_datetime(duk['date']).dt.date
del duk['date']
duk['count'] = 1
duk.groupby(['day','tags']).sum().sort_values('count', ascending=False)
```

Fig 19: Unusual activity country and day wise

Task 2: Characterising above days

Task 2.1: Making a word cloud

2.1.1: United Kingdom Word cloud

Fig 20: United Kingdom word cloud

2.1.2: Turkey Word cloud

Fig 21: Turkey word cloud

2.1.3: Italy Word cloud

Fig 22: Italy word cloud

Code logic and snippets:

All the texts from these unusual activities were fetched from text entity inside the data set and was stored inside a .txt file. Few basic data cleansing operations were performed on the text file to collect occurrences of unique words present in the text file and store it in the *dictionary*. Using *STOPWORDS, heapq, itemgetter* modules top 500 words were collected which excludes common words in particular language. Word cloud was generated using *wordcloud* module.

```
#Necessary libraries for this task
from wordcloud import WordCloud, STOPWORDS
import string
import heapq
from operator import itemgetter
from stop_words import get_stop_words

# creating a dictionary for occurences of unique words in text
d_uk = dict()
with open(temp1,"r",encoding='utf8') as text:
    for line in text:
        line = line.strip()
        line = line.lower()
        line = line.translate(line.maketrans("", "", string.punctuation))
        words = line.split(" ")
        for word in words:
            if word in d_uk:
                d_uk[word] = d_uk[word] + 1
            else:
                d_uk[word] = 1

#Fetching top 500 words and removing unnecessary and common words and
n=500
topitems = heapq.nlargest(n, d_uk.items(), key=itemgetter(1))
topitemsasdict = dict(topitems)
stopwords=set(STOPWORDS)
to_delete = set(topitemsasdict.keys()).difference(topitemsasdict.keys()-stopwords)
for d in to_delete:
    del topitemsasdict[d]

#Wordcloud generator
wc = WordCloud(background_color='white',
               width=1600,
               height=800,
               normalize_plurals = True,
               repeat = True
              )
wc.generate_from_frequencies(topitemsasdict)
wc.to_file(os.path.join(p1, 'uk_cloud.png'))
```

Fig 23: Word Cloud plotter

Task 2.2: Tree map representation

To characterise the events on the unusual high activity days and specific countries, Tree map was plotted.

2.2.1: United Kingdom Tree map

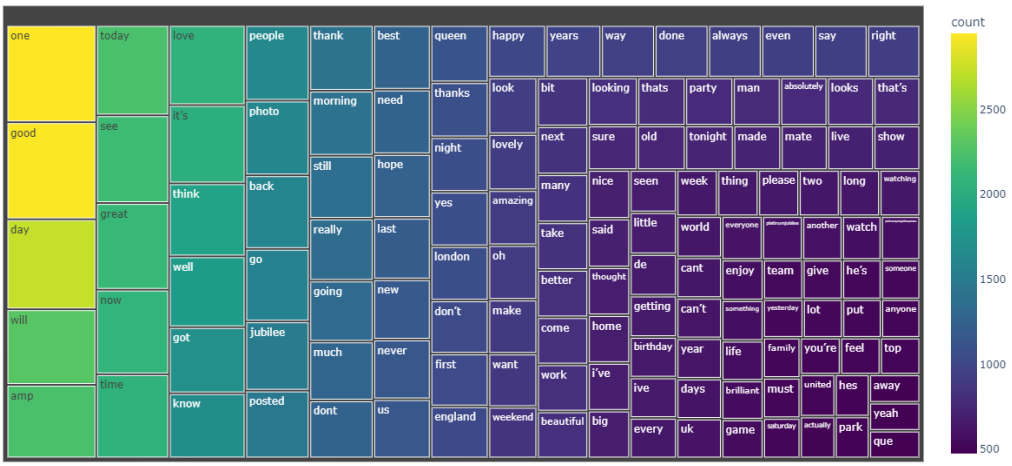


Fig 24: Tree map for UK

2.2.2: Turkey Tree map

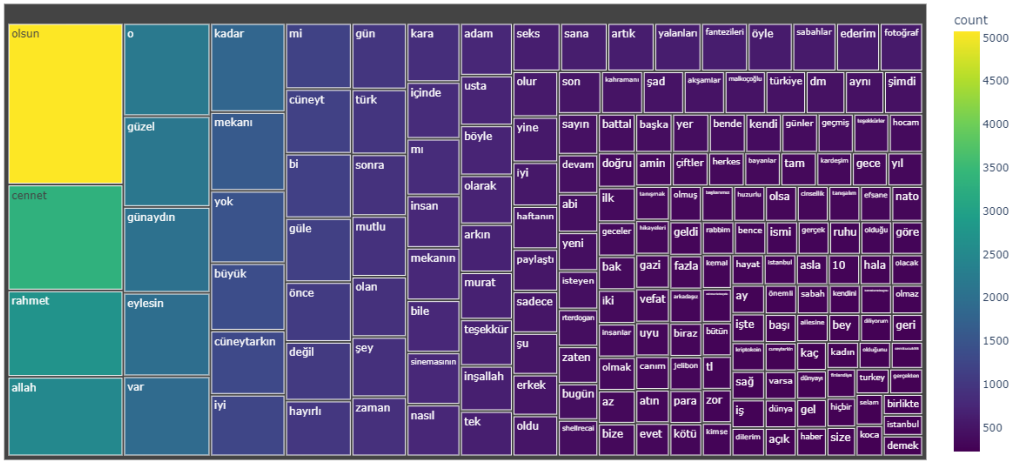


Fig 25: Tree map for Turkey

2.2.3: Italy Tree map

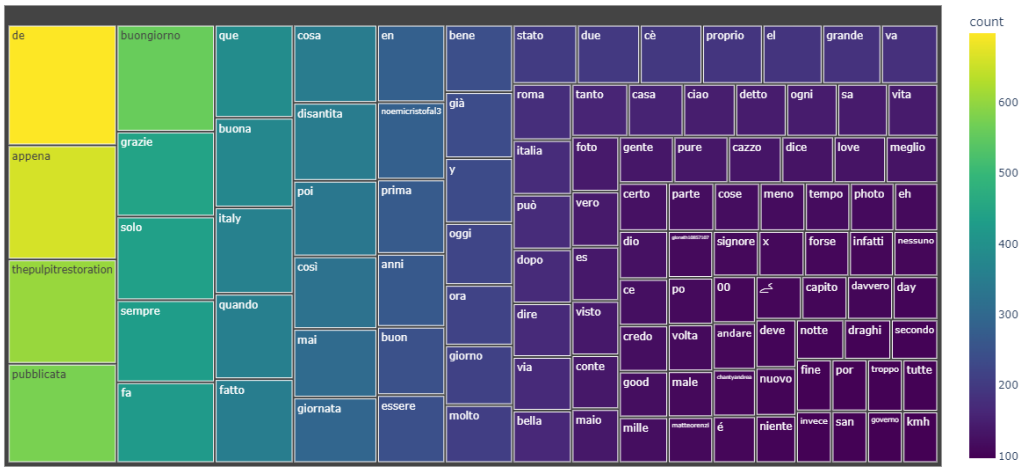


Fig 26: Tree map for Italy

Code logic and snippets:

Same dictionary from the above task was used as a data for plotting Tree map. Using *Treemap* from *plotly.express* module, the above desired Tree map was plotted.

```
import plotly.express as px
fig = px.treemap(topitemsasdict, path=['words'],values='count', width=1200, height=600,
                 color='count',color_continuous_scale='Viridis')
fig.update_layout(margin = dict(t=50, l=25, r=25, b=25))
fig.show()
```

Fig 27: Tree Map from Plotly express

Task 3: Summarise the events using data source

1. The Platinum Party at the Palace was a British music concert held on 4 June 2022 outside Buckingham Palace in London to commemorate Queen Elizabeth II's Platinum Jubilee. [3]
2. Fahrettin Cüreklibatır (7 September 1937 – 28 June 2022), better known by his stage name Cüneyt Arkın, was a Turkish movie actor, director, producer and martial artist. He was well-known for his roles in films such as The Mine, Dünyay Kurtaran Adam, and Paramparça.

Arkın died on June 28, 2022, at the age of 84, from cardiac arrest in a hospital in Istanbul, Turkey. [4]

3. This is related to a renovation of famous church called Kilpeck church which is in Herefordshire, Italy. It was a major restoration project in Italy during that time which accounted for a twitter trend [5]

Part 5: Reflection

Twitter and Twitter Data usage:

Social media, especially Twitter is main hub for all kinds of activities starting from promotions to news broadcast. In today's world Twitter is the first choice for dissemination in any field because of its reach, popularity and the way networking is made easy to users. Being a platform for mass online communications, it draws lot of attention from researchers, media and many more.

Uses and Misuses:

Firstly, there are lot of pros to using twitter data. Twitter with its worldwide connectivity and mass usage, have huge amount of information which can be used for several research like understanding customer voice, product review, political agendas, climatic information, and marketplace analysis when used in a proper and effective way. With various geography involved, it becomes a north star across the fields. Similarly, there are concerns with respect to mis usage of data as it involves personal information, propaganda, hatred, and inflammatory speech. It completely depends on an individual on how he uses the data and to what extent the ethical practices were met in terms of social media data analysis. Though it is a very complex and not so easy place to be to decide about rights and wrongs, it is bare minimum requirement for an individual to wield it subtly.

Drawbacks of Twitter data:

Twitter data is an open-source API, it contains information of tweeted user with their personal information if explicitly mentioned by the user in his profile. Data can be collected without users being aware that their tweets are being published/used. Users are rarely approached directly to obtain informed consent to participate in research; instead, consent is often assumed to have been granted by the user's acceptance of Twitter's Terms of Service. [6]

There is currently no agreement on how to proceed with this data, including the posting of specific posts According to a 2015 amendment to the United States' 'Common Rule' Federal Policy for the Protection of Human Subjects, certain types of online behaviour can be classified as public behaviour and thus do not require further ethical review. [7]

Ethical concerns around Twitter data:

Privacy breach, User consent for data usage, threat to a person basis his activity on twitter are few concerns that need to be addressed and handled in a very sensible way. Though there are ways to protect privacy of data and identity of user, it is not that easy task to adhere to ethics and other degrees of restrictions. Few cruxes related to ethical concerns are as follows:

1. Covering up usernames and @handles can avoid the identity of a person or a community, but it does not provide meaningful anonymity because entering the main text of a tweet into Twitter's search function can sometimes recover the tweet and its associated meta-data, including the user's username and @handle.
2. There is life threatening concern with respect to users' life which may come to harm if it is possible to identify them and they have been posting content considered to be hateful, inflammatory etc.

3. Twitter's User Development policy also says any form of reproduction or manipulation of tweets are breach of its policy and poses a risk. Hence, anonymisation is far behind for any kind of research. [8]
4. This requirement contradicts standard academic practises of anonymisation (although, as previously stated, this is difficult to achieve with tweets in any case) and also conflicts with European Union [9] and (in the UK) Information Commissioner's Office [10] data handling regulations.

Conclusion:

As it is very difficult to draw an agreement and follow a precise ethical behaviour by following all the norms, it should be dynamically done basis the kind of research undertaken and have a calculative approach towards the risk it poses. Considering the complexities and the scope for research in this field, addressing this in a big forum or debate is very significant and obliging for researchers.

References

[1] "Python Software Foundation," 23 11 2022. [Online]. Available: https://docs.python.org/3/library/datetime.html#strftime-strptime-behavior.

[2] "Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/List_of_European_countries_by_population. [Accessed 24 11 2022].

[3] Caroline Davies, "The Guardian," 02 06 2022. [Online]. Available: https://www.theguardian.com/uk-news/2021/jun/02/queens-platinum-jubilee-to-be-marked-with-four-day-bank-holiday-in-2022.

[4] "BBC News Turkey," BBC, 28 06 2022. [Online]. Available: https://www.bbc.com/turkce/haberler-turkiye-61963237.

[5] "Kilpeck Church Org," 22 06 2022. [Online]. Available: https://kilpeckchurch.org.uk/kilpeck-church-restoration-update-22-june-2022-review-of-major-works/.

[6] "Twitter T&Cs," Twitter, 10 06 2022. [Online]. Available: https://twitter.com/tos?lang=en%23us.

[7] "Office for Humar Research protections," Dept. of Health & Human services, U.S, 25 01 2017. [Online]. Available: https://www.hhs.gov/ohrp/regulations-and-policy/regulations/nprm-home/index.html.

[8] "Twitter Developer agreement & policy," Twitter, 10 10 2022. [Online]. Available: https://developer.twitter.com/en/developer-terms/agreement-and-policy.

[9] "Eur-Lex," European Union law, 24 05 2018. [Online]. Available: https://eur-lex.europa.eu/legal-content/en/TXT/?uri=CELEX:31995L0046.

[10] "Guide to data Protection," Information commissioner's office, Europe Union, 04 2021. [Online]. Available: https://ico.org.uk/for-organisations/guide-to-data-protection/.