Case Studies Assignment – 02

Riti Chakraborty(17231417)

Shreya Nagarkar(17231663)

Task 1 - Determining the trending policy topics in Galway based on analysis of Facebook pages or tweets from Local Media house (Newspaper or Radio station)

Accessing Facebook data

The Facebook data is fetched using Graph API explorer. Steps included are as follows:

1. Log in to: https://developers.facebook.com/.
2. After logging in we need to create an app which will give us access token and the secret key that will be required to fetch the data.
3. In this assignment we have used urllib to ping the URL that is formed using various attributes in order to fetch the appropriate data from Facebook.

Structure of the Query is as explained below:

https://graph.facebook.com/GalwayAdvertiser/posts/?fields=message,created_time,id&limit=100&access_token=##########################|##&since=20+oct+2017&until=23+march+2018&limit=100"

GalwayAdvertiser: this is the Galway local page from where the posts are fetched.

Posts: keyword used to fetch posts on the pages for e.g. feed is used to fetch feeds.

- Fields: contains message, created_time, and ID that we wish to collect.
- Limit=100: to limit the number of posts to be displayed in one page. Maximum 100 posts can be displayed at a time.
- Access_token: takes up the accesstoken|secretkey together for authentication purpose. The keys are hidden here for privacy issues.
- Since: and Until: is used to set the time period. As asked in the assignment the time period is set as 20+oct+2017 to 23+march+2018 i.e. past six months.
- For scraping the data a separate function is written FetchdataFB(). As mentioned above, only 100 posts are displayed at a time on the browser or the interface. The link to the next sets of data is mentioned in the paging section of the json file that is displayed. Thus, the function checks for the existence of the 'next' keyword in the page and pings the URL mentioned in the 'next' and the store the data into a .csv file.

After accessing the Facebook page and scraping the data from it the fetched data is stored in to a csv file 'facebook_page_statuses.csv' for further processing.

Identifying relevant posts from those obtained Analysis of the relevant posts to determine to provide required information (counts of posts per policy area)

- Step 1:

The first step used in this part of the assignment is to determine a pool of keywords belonging to specific government policies. The Policies chosen are Weather in Galway, Events in Galway and Accommodation in Galway. The Pool of these policy keywords are determined by analyzing a few open sources that give top keywords that are used in posts related to the mentioned policies. The Keywords of each file are chosen and stored as separate text files and are read in as pandas data frame. The files are 'one_Key.txt', 'housing_key.txt' and 'Events.txt' containing keywords for Weather Policy, Accommodation Policy and Events Policy respectively.

- Step 2:

After importing the determined keywords, we check the existence of these keywords in the posts and save the keywords that are present into a list so as to generate the set of relevant keywords and the set of relevant posts.

- Step 3:

After generating the set of relevant keywords, now we match the relevant keywords and the relevant posts to find the count if the posts that have the relevant keywords. The posts and the corresponding keywords are exported and stored in as a csv file named: 'Relevant_Post1.csv'.

- Step 4:

Finally, we create a data frame which contains the policy name of the relevant keywords, the keyword itself and the count of posts having those keywords. These statistics are store in to a csv file 'All.csv' top 20 keywords from these entire set are stored into the file called top_20 .csv. Top20 records displayed beside. From these two data frames, the visualizations are created.

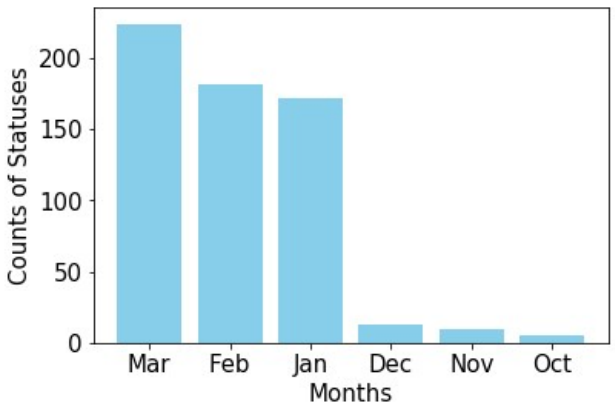
	Policy	Keywords	No_of_Posts_with_Keyword
10	Weather	air	37
1	Housing	rent	33
6	Weather	snow	32
0	Events	service	32
28	Weather	heat	31
2	Events	work	30
13	Weather	weather	28
15	Events	local	25
5	Housing	business	22
10	Housing	home	22
14	Events	services	20
15	Weather	rain	17
0	Housing	house	16
1	Events	match	16
18	Weather	flood	16
35	Weather	wind	16
31	Weather	hot	15
25	Events	full	15
9	Housing	properties	14

Visualization of the result

This step helps us to visualize the information in different forms such as graphs, charts etc. The visualizations created for this assignment are as shown below:

1. Number of posts per month:

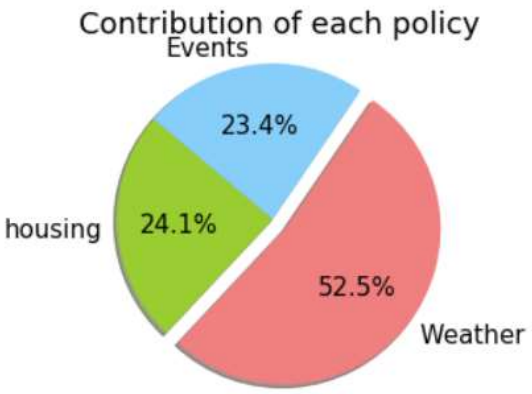
- The data fetched from Facebook has the posts for last 6 months
- The distribution of posts obtained per month varies.
- The total number of posts made on this page Galway Advertiser is as shown below with the help of a bar graph.



- From the graph above, it can be seen that the highest number of posts are in the month of March followed by February. October month was having the least count of posts.

2. Contribution of each Policy

- The distribution of posts fetched from the Facebook page is different for each policy.
- The pie chart below illustrates the contribution of each policy in the total posts
- It can be noted from the pie chart that relevant posts with keywords from weather policy are higher that housing followed by Events policies.

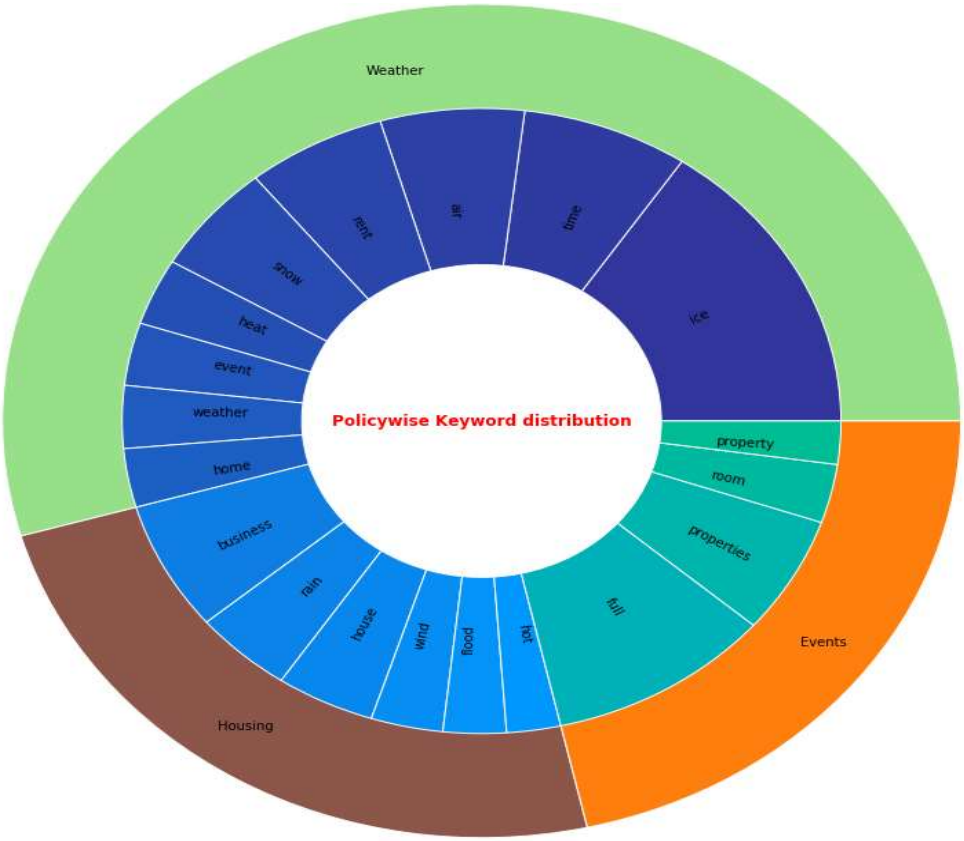


3. Multilayer donut representing the relevant keywords per policy and the count of posts with these keywords.

The donut below illustrates the contribution of each **"Relevant"** keyword towards the policy.

Selection of visual encodings:

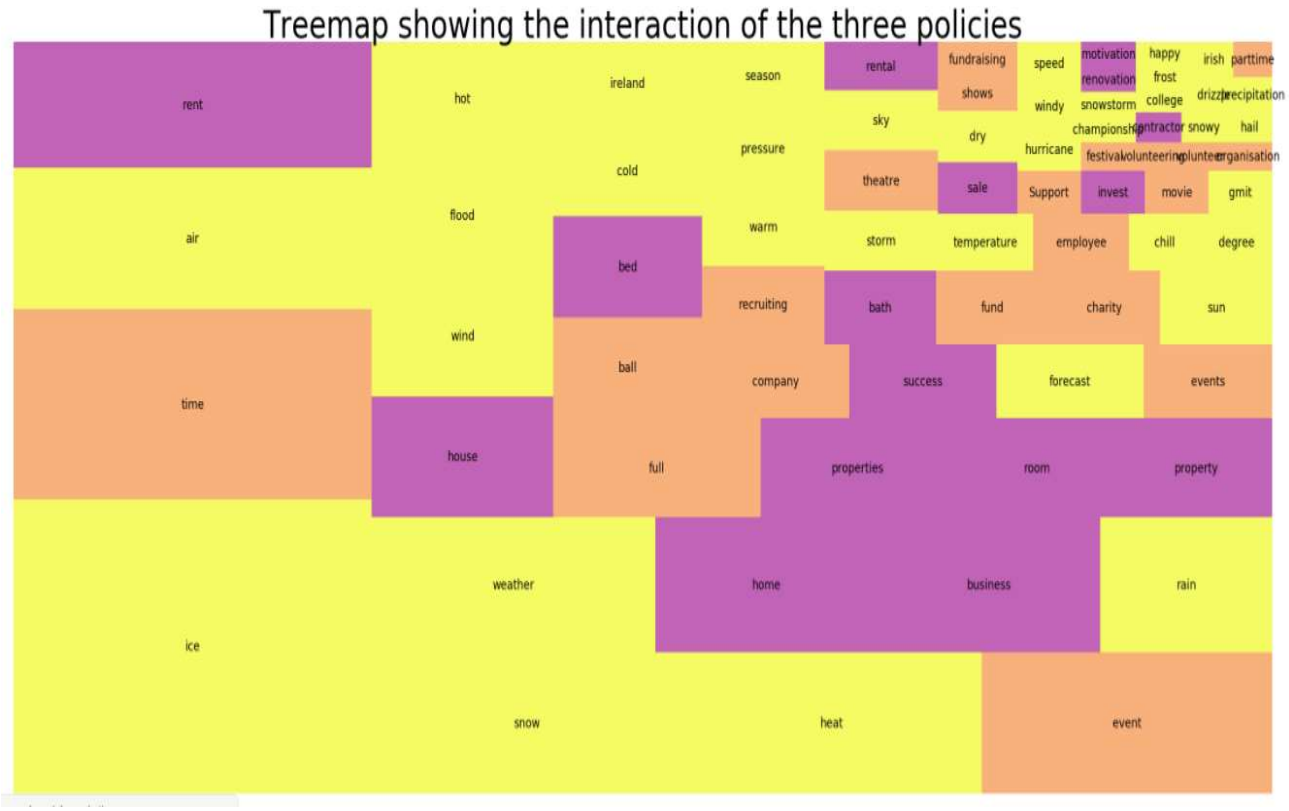
- The color for outer pie represents the policies that are chosen. For color coding, "terrain" palette is used.
- Green color is assigned to weather as the policy is more toward the nature whereas darker shade is assign to housing policy followed by orange to Events as events belong to certain happening measures.
- The inner pie color coding is chosen from "tab20" palette as it has more shades available which are assigned to keywords' contribution towards policy.
- The significance of the colors of the inner pie is, more the darker shade, more the contribution of the keyword. The size represents the total number of posts with keywords falling in that policy.



4. **<u>Tree map showing the interaction of all the three policies:</u>**

- All the keywords that are fetched here belong to certain policy, either Weather, Housing or Events.
- The Tree map below describes the belonging of words to the policy along with the contribution.
- The words in yellow box belong to the weather policy whereas words in pink to events and purple to housing policy.
- By looking at the Tree map, it can be concluded that, from the overall data fetched, more keywords belong to Weather policy followed by housing and events respectively.
- The size of the boxes represents the count of posts that fall under the respective category.

- For example, number of posts with the keyword ice in weather policy is more that the number of posts with the keyword air in the Weather Policy with the keyword rent in the Accommodation policy.



Conclusion and Observation:

1. It is observed that the keyword which are generally used to represent word respect to weather and the keywords used by general public to post on Facebook are indeed quite similar. This can be inferred from the charts represented above.
2. Also, the number of Relevant posts with Keyword form weather Policy is more in comparison to the other two.
3. It can also be observed for the other two policies i.e. housing and event, that the users tend to use more ambiguous to represent their thoughts.

References:

https://matplotlib.org/examples/color/colormaps_reference.html

https://github.com/minimaxir/facebook-page-post-scraper/blob/master/examples/how_to_build_facebook_scraper.ipynb

Appendix:

Code:

```
#Importing the different libraries
#for reading json file
import json

#for handling date and time
import datetime
import time
import calendar

#for reading and writing into csv file
import csv

#for requesting an URL and fetching data
import urllib.request

#for handling dataframes
import pandas as pd
from itertools import chain

#for treemap
import squarify
```

```
#for plots

import matplotlib.pyplot as plt

import plotly

import numpy as np


#Access token to fetch from Facebook (It is hidden for
privacy issues)

access_token="############"

#print(access_token)


#Check for occurence of exception. Sometime the URL doesnot
connect easily and gives error.

#This part of the code tries to reconnect to the server
until successful.

def request_until_succeed(url):

    req = urllib.request.Request(url)

    success = False

    while success is False:

        try:

            response = urllib.request.urlopen(req)

            #If the connection is successful the returned
code is 200 else 404

            if response.getcode() == 200:

                success = True

        except (Exception, e):

            time.sleep(5)

            print ("Error for URL %s: %s" % (url))
```

```
    return response.read()

# Formatting data before storing

def process_FB_Data(status):


    #formatting the message

    post_on_page = '' if 'message' not in status.keys()
else status['message'].encode('utf-8')



    #formatting the date

    page_published                                          =
datetime.datetime.strptime(status['created_time'],'%Y-%m-
%dT%H:%M:%S+0000')

    page_published       =       page_published       +
datetime.timedelta(hours=-5) # EST

    page_published  =  page_published.strftime('%Y-%m-%d
%H:%M:%S') # best time format for spreadsheet programs



    #returning results

    return (post_on_page, page_published)

#The main function to fetch the data from facebook.

def FetchdataFB(page_id):

    #Fecthing and storing the data into a CSV file

    with open('facebook_page_statuses.csv', 'w') as file:

        #Writing the data into the file

        w = csv.writer(file)

        next_page = True
```

```
        no_of_post_fetched = 0

        #Storing the URL with all the required fields.


url="https://graph.facebook.com/GalwayAdvertiser/posts/?f
ields=message,created_time,id&limit=100&access_token=2037
60660382790|4aPBk0cVP4ztPt-
q17f_l8K0xDQ&since=20+oct+2017&until=23+march+2018&limit=
100"

        #Data is fetched in the json format.

        data = json.loads(request_until_succeed(url))

        posts=data

        #Accessing the next page

        while next_page:

           #The data is stored in the form of Dictionary.
There going to the required level of the dictionary and
fetching the next link

           #Storing the data

            for i in posts['data']:

                w.writerow(process_FB_Data(i))

                no_of_post_fetched += 1

            #Checking for 'next' page links.

            if 'paging' in posts.keys():

                if 'next' in posts['paging'].keys():

                    next_link=posts['paging']['next']


                    #Calling   the   hadling   exception
function.

                    posts                                =
json.loads(request_until_succeed(next_link))
```

```
                else:
                    next_page = False
        #Printing total number of Posts fetched
        print (" %s Posts Fetched"%(no_of_post_fetched))


FetchdataFB(page_ID)


#Opening the File created above. It contains two fields: Posts and Created Time and Date
FB_data = pd.read_csv("facebook_page_statuses.csv",header=None)


#dropping the fields containing NA
FB_data=FB_data.dropna()


#renaming the columns
FB_data = FB_data.rename(columns={0:'Posts', 1:'DateTime'})


#Printing the Data accessed
print(FB_data)
#Defining Function to count relevant posts with respect to the policy keywords
def Count(Policy, filename):
    #For Storing the resulting relevant posts
    res_1=[]
```

```
    #For storing the corresponding keywords
    key_1=[]

    #Creating a Temporary array to store the keywords.
    temp_pool=[]

    #Creating a dictionary
    dict={}

    #It opens the file which is passed as an argument.
    with open(filename) as f:

        #Reading the contents of the file
        data = f.readlines()

        #Removing \n if it appears in the file of keywords
        line = [i.replace('\n','') for i in data]

        #Manually tokenizing the words and removing the
trailing spaces
        for i in line:
            #Storing the token into the temp_pool list
            temp_pool.append(i.rstrip().split(" "))

    #Storing the clean keywords.
```

```
    Keywords_pool=list(chain(*temp_pool))


    #Fetching the Posts from the Main Database which
contains the facebook data.

    Posts_list=list(FB_data['Posts'])


    #Defining empty list to store the matching keyword and
the matching post.

    match_keyword=[]

    relevant_post=[]


    #Checking for occurance of keyword in the posts.

    for i in Posts_list:

        for j in Keywords_pool:

            if j in i:

                #Storing tnhe matched Keywords and the
corresponding Posts

                match_keyword.append(j)

                relevant_post.append(i)


    #Removing Duplicates if present.

    relevant_post=set(relevant_post)

    match_keyword=set(match_keyword)


    #Defining empty lists.

    #Storing the Statistics of the keyword from the pool
of Relevant Keywpords.
```

```
    keyw=[]
    l=[]
    posts=[]
    posts_df=[]

    #Subsetting relevant posts and then storing them along
with the policy, count, keyword and the posts.
    for i in match_keyword:
        count=0
        tup=()
        for j in relevant_post:
            j.lower()
            if i in j:

                #Storing the relevant Posts
                #posts.append(j)
                count=count+1
                res_1.append(j)
                key_1.append(i)

        #creating a Tuple to store the Keyword and its
corresponding Count.
        tup=(i,count)

        #Copnverting the Tuple into a list
        l.append(tup)
```

```
    #Updating the Dictionary
    dict.update({Policy:l})


    #Creating a Dataframe of the Relevant Posts
    Result = pd.DataFrame({'Keyword': key_1,'Post': res_1})



    #Storing the relevant posts in a CSV file
    Result.to_csv('Relevant_Post1.csv', header=True, index=False)


    #Returning the Dictionary and the relevant posts
    return(dict,Result)



#Calling the function Created above which each of the keywords file


#Calling the functions with the policy of keywords in the weather policy
Weather_1=Count('Weather', 'one_Key.txt')
L = [(k, *t) for k, v in Weather_1[0].items() for t in v]
dframe1 = pd.DataFrame(L, columns=['Policy','Keywords','No_of_Posts_with_Keyword'])
dframe1=dframe1.sort_values('No_of_Posts_with_Keyword', ascending=False)
```

```
#Calling the functions with the policy of keywords in the
Housing policy

Housing_1=Count('Housing', 'housing_key.txt')

L = [(k, *t) for k, v in Housing_1[0].items() for t in v]

dframe2                  =                pd.DataFrame(L,
columns=['Policy','Keywords','No_of_Posts_with_Keyword'])

dframe2=dframe2.sort_values('No_of_Posts_with_Keyword',
ascending=False)


#Calling the functions with the policy of keywords in the
Events policy

Events_1=Count('Events', 'Events.txt')

L = [(k, *t) for k, v in Events_1[0].items() for t in v]

dframe3                  =                pd.DataFrame(L,
columns=['Policy','Keywords','No_of_Posts_with_Keyword'])

dframe3=dframe3.sort_values('No_of_Posts_with_Keyword',
ascending=False)


#Combining the rsults obtained above in one single
dataframe

frames=[dframe1,dframe2,dframe3]

final_table = pd.concat(frames)

final_table_sorted=final_table.sort_values('No_of_Posts_w
ith_Keyword',  ascending=False)

final_table_sorted=final_table_sorted[1:]

#Exporting into CSV file

final_table_sorted.to_csv('All.csv', header=True)
```

```
#Exporting top 20 records into csv file

final_table_sorted_top20=final_table_sorted.iloc[1:20,]

final_table_sorted_top20.to_csv('top_20.csv',
header=True)


#Printing top 20 records.

print(final_table_sorted_top20)

#Visualization

import matplotlib.pyplot as plt

# frames = [dframe1, dframe2[1:], dframe3]

# final_df = pd.concat(frames, ignore_index=True)

# final_df

# Group names is the list of policy names which is nothing
but the outer layer of the pie chart

group_names=['Weather', 'Housing', 'Events']


# The values in the below group_size list are the total
counts of keywords belonging to that particular policy

group_size=[269,119,106]


subgroup_names=final_table_sorted_top20['Keywords'].tolis
t()

# Subgroup size is the list of counts of keywords aranged
in the order of the policy

subgroup_size=[77,37,32,31,28,17,16,16,15,33,22,22,16,14,
12,50,30,15,11]
```

```
# Creating colors

# The colours are selection by selecting the pallate and
shade is selected by the number mentioned. More the number,
lighter the shade.

cm = plt.get_cmap("terrain")

cin                                                    =
cm(np.array([1,2,3,4,5,6,7,8,9,28,29,30,31,32,33,50,51,52
,53]))

cin                                                    =
cm(np.array([1,3,5,7,9,11,13,15,17,28,30,32,34,36,38,50,5
2,54,56]))



cm1 = plt.get_cmap("tab20")


# Outer pie chart

fig, ax = plt.subplots()

ax.axis('equal')

colors=cm1(np.array([5,10,2]))

mypie,      _       =      ax.pie(group_size,      radius=4,
labels=group_names, colors=colors,labeldistance=0.85)

plt.setp( mypie, width=1.5, edgecolor='white')


# Inner pie chart

mypie2,     _     =     ax.pie(subgroup_size,     radius=3,
labels=subgroup_names,labeldistance=0.65,
rotatelabels=90,colors=cin)

plt.setp( mypie2, width=1.5, edgecolor='white')

plt.margins(0,0)
```

```
kwargs = dict(size=12, fontweight='bold', va='center',
color="Red")

ax.text(0, 0, "Policywise Keyword distribution",
ha='center', **kwargs)


# Visualization

plt.show()

#for plotting a tree map for visualisation purpose, the
classes need to represented as integer value.

#Therefore we have randomly assigned each class to a random
number for visualising three different shades of color.

final_table_sorted_fr_treemap=final_table_sorted.replace(
{'Weather': 500, 'Events':188, 'Housing':94})



#Plotting the treemap for 'Weather Policy' with size as
counts of the keywords and label is corresponding keyword.

#Choosing color palatte

cmap = plt.cm.plasma

colors = [cmap(value) for value in
final_table_sorted_fr_treemap.Policy]


#Plotting the visualisation

plt.rc('font', size=22.5)

plt.figure(figsize=(45,25))



squarify.plot(sizes=final_table_sorted_fr_treemap["No_of_
Posts_with_Keyword"],
```

```
label=final_table_sorted_fr_treemap["Keywords"],
color=colors, alpha=0.7)


# Mentioning the title of the plot with size

plt.title("Treemap showing the interaction of the three
policies", size=40)


plt.axis('off')


plt.show()

# From the dataframe of 'Housing' policy, the rows having
0 counts are dropped using below code.

ph=[]

for i in range(0,len(dframe2)):

        if((dframe2['No_of_Posts_with_Keyword'][i])==0):

            ph.append((dframe2.index[i]))


# dk is the name of dataframe with 0 values dropped.

for k in ph:

    dk=dframe2.drop(dframe2.index[ph])


Count_Weather = sum(dframe1['No_of_Posts_with_Keyword'])

l1=list(dframe1['No_of_Posts_with_Keyword'])

Count_Housing=sum(dframe2['No_of_Posts_with_Keyword'][1:]
)

l2=list(dframe2['No_of_Posts_with_Keyword'])
```

```
# The date formart in the CSV file is YYYY-MM-DD time. To
get the month and corresponding name, the folloing code is
written

# Fetching date&Time column and converting to list.

f1=FB_data['DateTime'].tolist()

list_month=[]

month_name=[]

month_name_short=[]


# Fetching the digits of the month

for i in range(0,len(f1)):

    list_month.append(f1[i][5:7])


#The count of tweets for each month is saved in dictionary
with month as key and corresponding count as value

dc = {x:list_month.count(x) for x in list_month}

new_list=[]

for i in dc.keys():

    new_list.append(i)


# Month name is calculated using calendar.month_name
function.

for j in range(0,len(new_list)):

    n=int(new_list[j])

    month_name.append(calendar.month_name[n])


# Substring month names
```

```
for k in range(0,len(month_name)):
    month_name_short.append(month_name[k][0:3])
# Plotting the monthwise count of posts
#Mentioning the values of X&y axis
plt.bar(range(len(dc)),    dc.values(),    align='center',
color="skyblue")
plt.xticks(range(len(dc)), month_name_short)

# Mentioning labels and figure size
plt.ylabel('Counts of Statuses')
plt.xlabel('Months')

#Displaying the graph
plt.show()
new_dic={}
s_housing=sum(dframe2['No_of_Posts_with_Keyword'][1:])
new_dic.update({'housing':s_housing})
s_Weather=sum(dframe1['No_of_Posts_with_Keyword'])
new_dic.update({'Weather':s_Weather})
s_events=sum(dframe3['No_of_Posts_with_Keyword'])
new_dic.update({'Events':s_events})
# Data Visualization of contribution of policies

# Importning required libraries
import plotly.plotly as py
import plotly.graph_objs as go
```

```
# The label od pie chart will be the keys that is name of
policy taken from dictionary & Values are the corresponding
counts

labels = list(new_dic.keys())

values = list(new_dic.values())

# Explode is used to slice the chart. The lower the value
of explode, lower the distance of the slice from the chart

explode = (0.0, 0.1, 0)

# Mentioning the colours of the pie chart

colors = ['yellowgreen', 'lightcoral', 'lightskyblue']


# Plotting the chart

plt.pie(values,      explode=explode,      labels=labels,
colors=colors,

        autopct='%1.1f%%', shadow=True, startangle=140)

# Mentioning the title

plt.title("Contribution of each policy")

plt.axis('equal')

plt.show()
```