```
from google.colab import drive
drive.mount('/content/drive')
%cd /content/drive/MyDrive/Precog

     Mounted at /content/drive
     /content/drive/MyDrive/Precog
```

```
import nltk
nltk.download('stopwords')

     [nltk_data] Downloading package stopwords to /root/nltk_data...
     [nltk_data]   Package stopwords is already up-to-date!
     True
```

References

https://www.dezyre.com/student-project/toly-novik-text-mining-and-clustering-of-tweets-based-on-context/2#:~:text=Identify%20tweets%20that%20are%20talking,used%20words%20in%20each%20topic.

Extract tweets

```
# !python3 tw.py
```

Imports

```
import pandas as pd
import numpy as np
import re

# plot
# import seaborn as sns
import matplotlib.pyplot as plt

# Gensim for sentiment analysis
import gensim
from gensim.utils import simple_preprocess

import nltk
from nltk.corpus import stopwords, twitter_samples

from collections  import Counter
from wordcloud import WordCloud
from wordcloud import WordCloud

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

import tweepy
from tweepy import OAuthHandler
from textblob import TextBlob

from tqdm import tqdm
import matplotlib.pyplot as plt
```

Preprocessing

```
df = pd.read_csv('./df.csv')
```

```
df = df.dropna()
df.head()
```

	CreatedAt	tweets	User_ID	User_Name
0	2020-12-28 14:09:51	b'RT @TicketNew: \xf0\x9f\x9a\xa8\xf0\x9f\x94\...	1.189225e+18	Raj🅰☮
2	2020-12-28 14:09:50	b'@Aji_spartan \n\n@sharan14110326 \n\n@Vasant...	1.251350e+18	💘Jeni💘Ice Doll♥LK
4	2020-12-28 14:09:46	b"RT @ManokarVj: Teaser &amp; Trailer announce...	1.008243e+18	Bigil Ashok

```
df.rename(columns = {'Tweet': 'tweets'}, inplace=True)
```

```
count = df['tweets'].str.split().str.len()
count.index = count.index.astype(str)+' words:'
print("Total number of words in the tweets =",count.sum())
print("Mean number of words per tweet :", round(count.mean(), 2))
```

```
Total number of words in the tweets = 186033
Mean number of words per tweet : 13.68
```

```
print('Users: ', len(df['User_ID'].unique()))
df['tweet_length'] = df['tweets'].str.len()
print("total characters in tweets =", df['tweet_length'].sum())
print("mean number of characters per tweet =", df['tweet_length'].mean())
```

```
df = df.drop(['tweet_length'], axis=1)
```

```
     Users:  3678
     total characters in tweets = 2242975
     mean number of characters per tweet = 164.9246323529412
```

```
def remove_users(tweet, pattern1, pattern2):
        r = re.findall(pattern1, tweet)
        for i in r:
            tweet = re.sub(i, '', tweet)
        r = re.findall(pattern2, tweet)
        for i in r:
            tweet = re.sub(i, '', tweet)
        return tweet
```

```
df['tidy_tweets'] = np.vectorize(remove_users)(df['tweets'],"@ [\w]*", "@[\w]*")
```

```
df['tidy_tweets']=df['tidy_tweets'].str.lower()
df['tidy_tweets']=df['tidy_tweets'].apply(lambda x: x.strip('b').strip("'").strip('"').replac

# has hashtags so removing them
df['tidy_tweets'] = np.vectorize(remove_users)(df['tidy_tweets'], "# [\w]*", "#[\w]*")

filtr = df['tidy_tweets'].str.len() != 0
df = df[filtr]
```

```
def remove_links(tweet):
    tweet_no_link = re.sub(r"http\S+", "", tweet)
    return tweet_no_link
```

```
# removing links

df['tidy_tweets'] = np.vectorize(remove_links)(df['tidy_tweets'])

# removing punctuations
df['tidy_tweets'] = df['tidy_tweets'].str.replace("[^a-zA-Z#]", " ")

# removing shortwords
df['tidy_tweets'] = df['tidy_tweets'].apply(lambda x: ' '.join([i for i in x.split() if len(i
```

```
def tokenize(tweet):
    for word in tweet:
        yield(gensim.utils.simple_preprocess(str(word), deacc=True))
```

```
df['tidy_tweet_tokens'] = list(tokenize(df['tidy_tweets']))
```

```
def remove_stopwords(tweets):
    return [[word for word in simple_preprocess(str(tweet)) if word    not in stop_words] for


# now remove stopwords
stop_words = stopwords.words('english')
stop_words.extend(['from', 'https', 'twitter', 'religions','pic','twitt',])
df['tokens_no_stop'] = remove_stopwords(df['tidy_tweet_tokens'])
print("\nSTOPWORDS REMOVED\n")
print(df['tokens_no_stop'].head())

# REMOVE TWEETS LESS THAN 3 TOKENS
df['length'] = df['tokens_no_stop'].apply(len)
df = df.drop(df[df['length']<3].index)
df = df.drop(['length'], axis=1)
```

```
STOPWORDS REMOVED

0                                    [tomorrow, ready]
2                               [follow, keep, suppor]
4    [teaser, trailer, announcement, pics, thalapat...
6                    [chances, falling, sunday, papom]
8                                      [update, today]
Name: tokens_no_stop, dtype: object
```

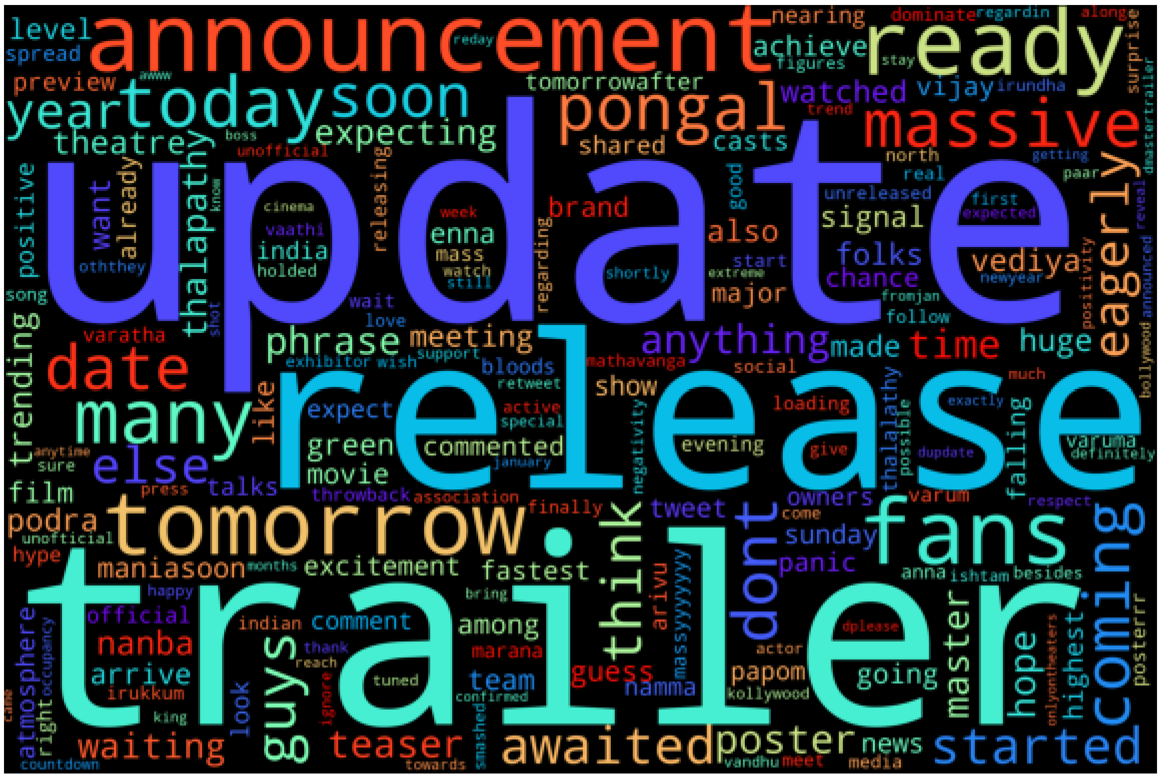
```
!pip install tqdm
```

```
Requirement already satisfied: tqdm in /usr/local/lib/python3.6/dist-packages (4.41.1)
```

```
from tqdm import tqdm
# generating a wordcloud for visual representation
string=''
for i,r in tqdm(df.iterrows()):
    string+=' '.join(r['tokens_no_stop'])+' '

# wordcloud = WordCloud(width = 800, height = 800,
#                 background_color ='white',
#                 stopwords = stop_words,
#                 min_font_size = 10).generate(string)

wordcloud = WordCloud(width = 3000, height = 2000, random_state=1, background_color='black',
# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```

```
wordcloud.to_file('./wordcloud.png')
df.to_pickle('./pre-processed-tweets.pkl')
```

```
9691it [00:00, 10848.41it/s]
```



Sentiment Analysis

```
df = pd.read_pickle('pre-processed-tweets.pkl')
```

```
df.head()
```

	CreatedAt	tweets	User_ID	User_Name	tidy_tweets	tidy_tw
2	2020-12-28 14:09:50	b'@Aji_spartan \n\n@sharan14110326 \n\n@Vasant...	1.251350e+18	🤍Jeni🤍Ice Doll♥LK	follow keep suppor	[
4	2020-12-28 14:09:46	b"RT @ManokarVj: Teaser &amp; Trailer announce...	1.008243e+18	Bigil Ashok	teaser trailer announcement pics thalapathy vi...	[te announc
6	2020-12-28 14:09:40	b'RT @dp_karthik09: Chances of #MasterTrailer ...	9.701324e+17	Saravana Pradeep	chances falling sunday papom	[cha sur

```
def get_tweet_sentiment(tweet):
  '''
  Utility function to classify sentiment of passed tweet
  using textblob's sentiment method
  '''
  # create TextBlob object of passed tweet text
  analysis = TextBlob(tweet)

  # print(analysis.sentiment)

  # set sentiment
  if analysis.sentiment.polarity > 0:
    return 'positive', analysis.sentiment.polarity, analysis.sentiment.subjectivity

  elif analysis.sentiment.polarity == 0:
    return 'neutral', analysis.sentiment.polarity, analysis.sentiment.subjectivity
  else:
    return 'negative', analysis.sentiment.polarity, analysis.sentiment.subjectivity



'''
calculating polarities for tweets on the given hashtag

link to output of the cell :
https://colab.research.google.com/drive/1B1KqIjwpTMgC8L1C0eooeUNscjkQXdua?authuser=1#scrollTo

'''
from tqdm import tqdm
sentiments = []
polarities = []
subjectivities = []
for i,r in tqdm(df.iterrows()):
  temp = get_tweet_sentiment(r['tidy_tweets'])
  sentiments.append(temp[0])
  polarities.append(temp[1])
  subjectivities.append(temp[2])
df['sentiment'] = sentiments
df['polarity'] = polarities
df['subjectivity'] = subjectivities
df.head()

"""
The below line was used to save the modified dataframe
Uncomment to do save the dataframe with polarities.
"""
# df['User_ID'] = df['User_ID'].astype('int64')
# df.to_csv("./sentiments_and_polarities_with_pre_processed.csv", index=False)
```

```
     9691it [00:06, 1433.25it/s]
```

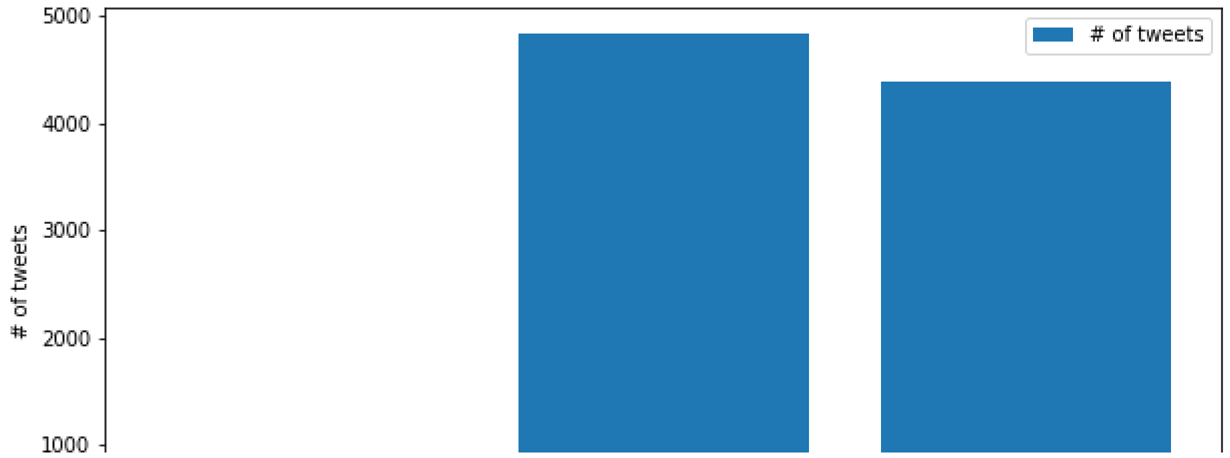
```
df = pd.read_csv('./sentiments_and_polarities_with_pre_processed.csv')


# getting unique users
user = df['User_ID'].unique()
print("Number of unique users in the sampled dataset",len(user))

     Number of unique users in the sampled dataset 3026


# Creating a dictionary which stores number of tweets of a user on "#MasterTrailer"
user_freq = dict(zip(user, [0]*len(user)))
for i,r in df.iterrows():
  user_freq[r['User_ID']]+=1
#this shows how active a user was


'''
The below code was just used to save the users' active status.
Uncomment to do save the findings.
'''
# file = open("unique_users.txt", "w+")
# for i in user:
#   file.write(str(i))
#   file.write("\n")
# file.close()

# file = open("users_freq.txt", "w+")
# file.write(str(user_freq))
# file.close()


# [negative, neutral, positive]
df = pd.read_csv('sentiments_and_polarities_with_pre_processed.csv')
sentiment_of_user=dict(zip(user, [0]*len(user)))
for i,r in tqdm(df.iterrows()):
  polarity = r['polarity']
  if type(sentiment_of_user[r['User_ID']])==int:
    sentiment_of_user[r['User_ID']]=[0,0,0]
  if polarity>0:
    sentiment_of_user[r['User_ID']][2]+=1
  elif polarity==0.0:
    sentiment_of_user[r['User_ID']][1]+=1
  else:
    sentiment_of_user[r['User_ID']][0]+=1

'''
The below code is just to save our findings
Uncomment to do save the dictionary.
'''
# file = open("sentiments_of_users.txt", "w+")
# file.write(str(sentiment_of_user))
# file.close()
```

```
9691it [00:01, 9485.62it/s]
'\nThe below code is just to save our findings\nUncomment to do save the dictionary.\n'
```

```
req_list = list(sentiment_of_user.items())
req_list.sort(key=lambda x: x[0])

y1 = [i[1][0] for i in req_list] #negative
y2 = [i[1][1] for i in req_list] #neutral
y3 = [i[1][2] for i in req_list] #positive

x1 = [i[0] for i in req_list] # users
x2 = [i[0] for i in req_list] # users
x3 = [i[0] for i in req_list] # users

plt.figure(figsize=(10,5))
plt.bar(list(range(len(req_list))),y1, label="Neg", width=5)

# plt.bar(left, height, tick_label = tick_label,
#         width = 0.8, color = ['red', 'green'])

plt.xlabel("User's Index in req_list")
plt.ylabel("# negative tweets by the user")

plt.legend()
# plt.savefig('plot_neg.png')
plt.show()

plt.figure(figsize=(10,5))
plt.bar(list(range(len(req_list))),y2, label="Neu", width = 5)
plt.xlabel("User's Index in req_list")
plt.ylabel("# neutral tweets by the user")

plt.legend()
# plt.savefig('plot_neu.png')
plt.show()

plt.figure(figsize=(10,5))
plt.bar(list(range(len(req_list))),y3, label="Pos")
plt.xlabel("User's index in the req_list")
plt.ylabel("# positive tweets by the user")

plt.legend()
# plt.savefig('plot_pos.png')
plt.show()
```

```
req_list = list(sentiment_of_user.items())
req_list.sort(key=lambda x: x[0])

neg=pos=neu=0
```

```
  for i in req_list:
    neg+=i[1][0]
  for i in req_list:
    neu+=i[1][1]
  for i in req_list:
    pos+=i[1][2]

  x1 = ['negative', 'neutral', 'positive'] # users

  plt.figure(figsize=(10,5))
  plt.bar(x1,[neg, neu, pos], label="# of tweets")
  plt.xlabel("Sentiment of tweets")
  plt.ylabel("# of tweets")

  plt.legend()
  plt.savefig('plot_tweets_sentiments.png')
  plt.show()

  # Pie chart, where the slices will be ordered and plotted counter-clockwise:
  # labels = ['Same Polarities', 'Different Polarities']
  # sizes = [same_polarities, different_polarities]
  explode = (0.1, 0, 0)

  fig1, ax1 = plt.subplots()
  ax1.pie([neg, neu, pos], explode=explode, labels=x1, autopct='%1.1f%%',
          shadow=True, startangle=90)
  ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
  plt.title("Distribution of polarities among tweets ")
  plt.savefig('plot_tweets_pie.png')
  plt.show()
```



Friends

```
'''
To get the tweets of friends of the users on the "#MasterTrailer"
Uncomment below line
'''
# !python3 get_friends.py
# !python3 get_friends_tweets.py
```

```
df_friends_tweets = pd.read_csv('friend_tweets.csv')
df_friends_tweets.head()
```

	user_id	friend_id	created_at	text
0	1039062325645328384	1344621554550890498	2020-12-31 12:28:55	#MasterTrailer release date Poster ?
1	1039062325645328384	1344588889629970433	2020-12-31 10:19:07	Keep Calm and be Don't Confused 👍 #MasterTrail...
2	1039062325645328384	1344584551918567426	2020-12-31 10:01:53	#MasterTrailer Holds Whole Movie Cast &amp; Crew😡

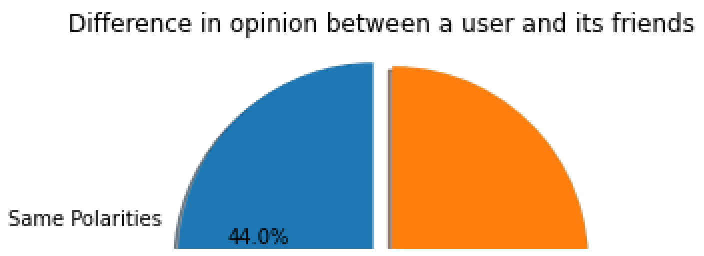
```
from tqdm import tqdm
# df['sentiment'] = ""
# df['polarity'] = None
# df['sentiment'].apply(get_tweet_sentiment)
sentiments = []
polarities = []
subjectivities = []
for i,r in tqdm(df_friends_tweets.iterrows()):
  temp = get_tweet_sentiment(r['text'])
  sentiments.append(temp[0])
  polarities.append(temp[1])
  subjectivities.append(temp[2])
df_friends_tweets['sentiment'] = sentiments
```

```
df_friends_tweets['polarity'] = polarities
df_friends_tweets['subjectivity'] = subjectivities

df_friends_tweets.head()
```

```
141it [00:00, 897.66it/s]
```

	user_id	friend_id	created_at	text	sentiment
0	1039062325645328384	1344621554550890498	2020-12-31 12:28:55	#MasterTrailer release date Poster ?	neutral
1	1039062325645328384	1344588889629970433	2020-12-31 10:19:07	Keep Calm and be Don't Confused 👍 #MasterTrail...	negative

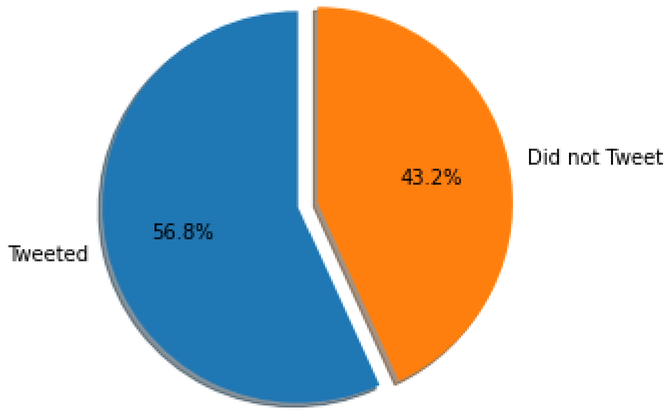
```
'''
user_friend_dict has (user_id, [polarity_sum, subjectivity_sum]) as (Key,Value) pair
polarity_sum is nothing but sum of the polarities of tweets retrieved of the friends of key (
subjectivity_sum is nothing but sum of the subjectivities of tweets retrieved of the friends
'''

temp_friends = df_friends_tweets.groupby('user_id')
user_friend_dict = {}
for i,r in temp_friends:
  polarity = r['polarity'].sum()/r['polarity'].count()
  subjectivity = r['subjectivity'].sum()/r['polarity'].count()
  user_friend_dict[i]=[polarity, subjectivity]

user_friend_dict


user_polarity = {}
for i, r in df.groupby('User_ID'):
  polarity = r['polarity'].sum()/r['polarity'].count()
  subjectivity = r['subjectivity'].sum()/r['polarity'].count()
  user_polarity[i]=[polarity, subjectivity]
user_polarity


users_list = []
friends_list = []
for k in user_friend_dict.keys():
  users_list.append(user_polarity[k])
  friends_list.append(user_friend_dict[k])
same_polarities = different_polarities = 0

for i in range(len(users_list)):
  if users_list[i][0]>0.0 and friends_list[i][0]>0.0:
    same_polarities+=1
  elif users_list[i][0]==0.0 and friends_list[i][0]==0.0:
    same_polarities+=1
  elif users_list[i][0]<0.0 and friends_list[i][0]<0.0:
    same_polarities+=1
  else:
    different_polarities+=1


# Pie chart, where the slices will be ordered and plotted counter-clockwise:
labels = ['Same Polarities', 'Different Polarities']
sizes = [same_polarities, different_polarities]
explode = (0, 0.1)

fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.title("Difference in opinion between a user and its friends")
plt.savefig('plot_users_friends_polarities.png')
plt.show()

plt.figure(figsize=(10,5))
plt.bar(labels,sizes, label="# of users")
plt.xlabel("Sentiment")
plt.ylabel("# of users")
plt.legend()
plt.show()
```

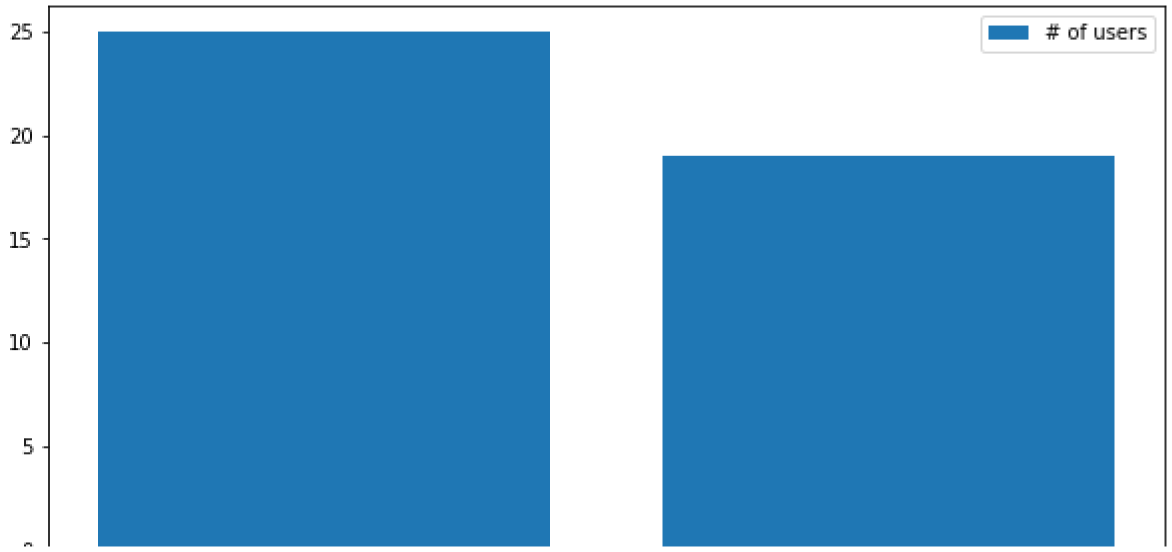


```
friend_count_dict = {}
file = open('./list_friends.txt', "r")
lines = file.readlines()
for i in tqdm(range(0,len(lines),2)):
    user, followers = lines[i], eval(lines[i+1])
    followers = followers[:20]
    friend_count_dict[int(user)] = [0,len(followers)]

for i,r in df_friends_tweets.groupby('user_id'):
  friend_count_dict[i][0]=r.shape[0]

no_tweet_by_friends = 0

for i in friend_count_dict.keys():
  if friend_count_dict[i][0]==0:
    no_tweet_by_friends += 1


# friend_count_dict



# Pie chart, where the slices will be ordered and plotted counter-clockwise:
labels = ['Tweeted', 'Did not Tweet']
sizes = [len(list(friend_count_dict.keys()))-no_tweet_by_friends, no_tweet_by_friends]
explode = (0, 0.1)

fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.title("Difference in the number of users with friends who tweeted regaring the same hasht
plt.savefig('plot_users_friends_tweeted.png')
plt.show()

plt.figure(figsize=(10,5))
plt.bar(labels,sizes, label="# of users")
# plt.xlabel("Sentiment")
# plt.ylabel("# of users")
plt.legend()
plt.show()
```

```
100%|██████████| 44/44 [00:00<00:00, 558.41it/s]
```

Difference in the number of users with friends who tweeted regaring the same hashtag



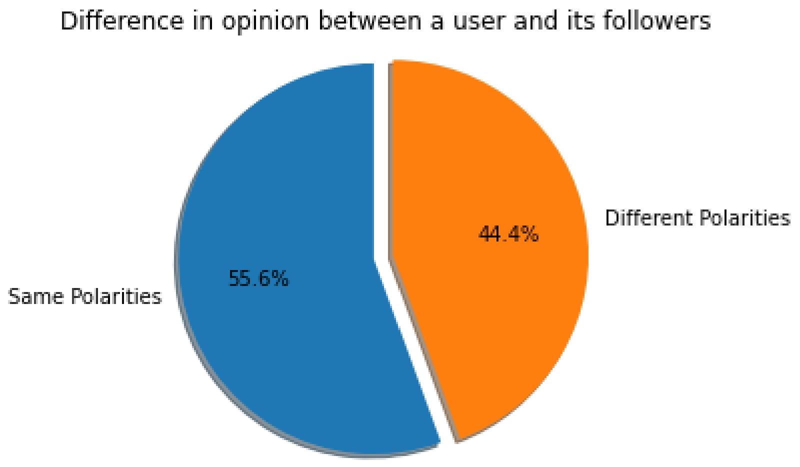


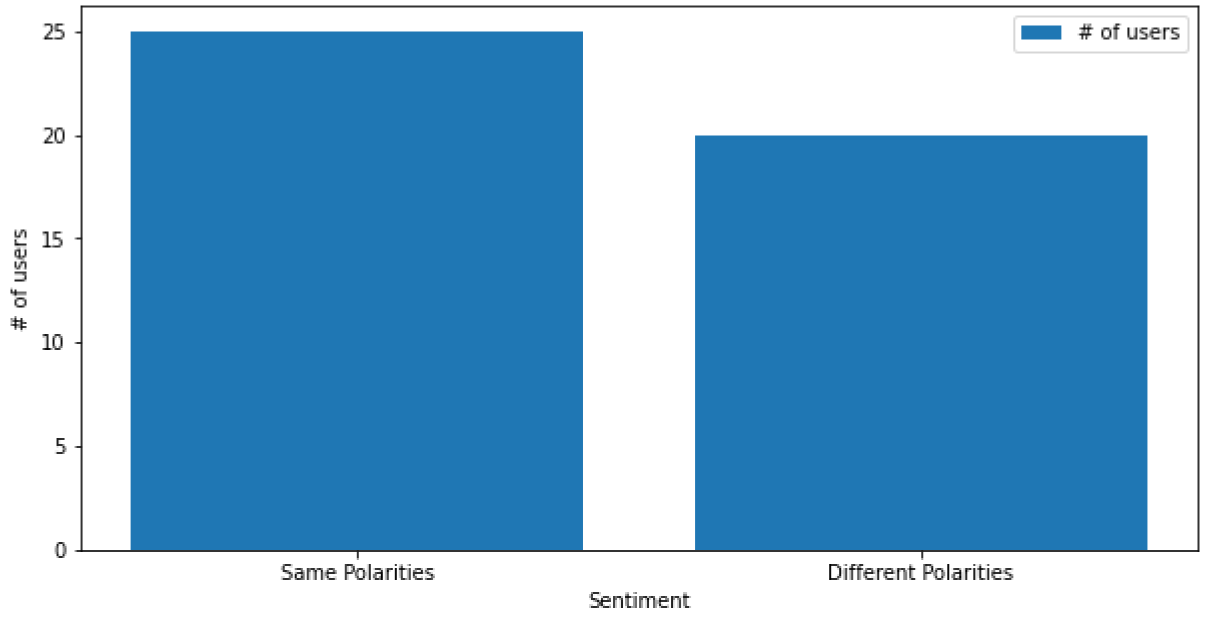
Followers

```
'''
To extract followers of users and their respective tweets with the hashtag "#MasterTrailer"
Uncomment the below lines
'''
# !python3 get_followers.py
# !python3 get_followers_tweets.py


df_followers_tweets = pd.read_csv('follower_tweets.csv')


from tqdm import tqdm
sentiments = []
polarities = []
subjectivities = []
for i,r in tqdm(df_followers_tweets.iterrows()):
  temp = get_tweet_sentiment(r['text'])
  sentiments.append(temp[0])
  polarities.append(temp[1])
```

```
    subjectivities.append(temp[2])
  df_followers_tweets['sentiment'] = sentiments
  df_followers_tweets['polarity'] = polarities
  df_followers_tweets['subjectivity'] = subjectivities

     344it [00:00, 1290.99it/s]


  user_follower_dict = {}
  for i,r in df_followers_tweets.groupby('user_id'):
    polarity = r['polarity'].sum()/r['polarity'].count()
    subjectivity = r['subjectivity'].sum()/r['polarity'].count()
    user_follower_dict[i]=[polarity, subjectivity]

  user_follower_dict


  users_list = []
  followers_list = []
  for k in user_follower_dict.keys():
    users_list.append(user_polarity[k])
    followers_list.append(user_follower_dict[k])
  same_polarities = different_polarities = 0

  for i in range(len(users_list)):
    if users_list[i][0]>0.0 and followers_list[i][0]>0.0:
      same_polarities+=1
    elif users_list[i][0]==0.0 and followers_list[i][0]==0.0:
      same_polarities+=1
    elif users_list[i][0]<0.0 and followers_list[i][0]<0.0:
      same_polarities+=1
    else:
      different_polarities+=1


  # Pie chart, where the slices will be ordered and plotted counter-clockwise:
  labels = ['Same Polarities', 'Different Polarities']
  sizes = [same_polarities, different_polarities]
  explode = (0, 0.1)

  fig1, ax1 = plt.subplots()
  ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
          shadow=True, startangle=90)
  ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
  plt.title("Difference in opinion between a user and its followers")
  plt.savefig('plot_users_followers_polarities.png')
  plt.show()

  plt.figure(figsize=(10,5))
  plt.bar(labels,sizes, label="# of users")
  plt.xlabel("Sentiment")
  plt.ylabel("# of users")
```

```
plt.legend()
plt.show()
```





```
follower_count_dict = {}
file = open('./list_followers.txt', "r")
lines = file.readlines()
for i in tqdm(range(0,len(lines),2)):
    user, followers = lines[i], eval(lines[i+1])
    followers = followers[:20]
    follower_count_dict[int(user)] = [0,len(followers)]

for i,r in df_followers_tweets.groupby('user_id'):
  follower_count_dict[i][0]=r.shape[0]

no_tweet_by_followers = 0

for i in follower_count_dict.keys():
  if follower_count_dict[i][0]==0:
    no_tweet_by_followers += 1
```

```
# friend_count_dict


# Pie chart, where the slices will be ordered and plotted counter-clockwise:
labels = ['Tweeted', 'Did not Tweet']
sizes = [len(list(follower_count_dict.keys()))-no_tweet_by_followers, no_tweet_by_followers]
explode = (0, 0.1)

fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.title("Difference in the number of users with followers who tweeted regarding the same ha
plt.savefig('plot_users_followers_tweeted.png')
plt.show()

plt.figure(figsize=(10,5))
plt.bar(labels,sizes, label="# of users")
# plt.xlabel("Sentiment")
# plt.ylabel("# of users")
plt.legend()
plt.show()
```

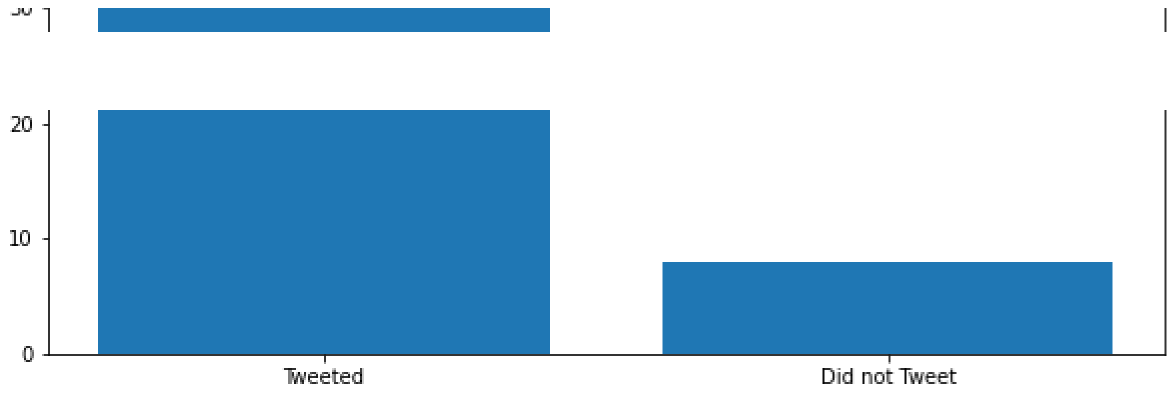
```
100%|██████████| 53/53 [00:00<00:00, 700.70it/s]
```

Difference in the number of users with followers who tweeted regarding the same hashtag
Did not Tweet

Report

The above analysis shows a wide range of factor to compare and contrast the users, their followers and their friends.

But this is not just it. A lot more could be done with the provided data. I fell short on time, else Latent Dirichlet Allocation (LDA) could be used for topic modelling the users' tweets and we can get an in-depth insight into the tweets. Moreover, I have used TextBlob library of Python for sentiment analysis. Rather a better way to do it was writing a model for sentiment analysis from scratch where Long Short Term Memory(LSTM) could be used and fine-tuned to our necessity.


Tweeted
Did not Tweet