Importing Necessary Libraries

We will be using :

1) **Regex(re)** to extract and manipulate strings based on specific patterns

2) **Pandas** for analysis

3) **Matplotlib** for visualization

4) **Emoji** to deal with emojis

import re  
import pandas as pd  
import matplotlib.pyplot as plt  
import emoji

From Where To Obtain Data

Step 1: Go to your Whatsapp Chat you want to analyze

Step 2: Tap on the 3 dots present on the top right corner

Step 3: Choose Export Chat option & mail it to yourself

Step 4: Download and Start Exploring

file = open(r'./WhatsApp Chat with Manish AG.txt',mode='r',encoding="utf8")  
data = file.read()  
file.close()  
data

*Note: While exporting data avoid including media files because if the number of media files is greater than certain figure then not all media files are exported.*

After storing our chats in our **data**variable we will the format in which chat is available

Our data looks something like this:

12:24 - Suyash: Hey\n03/01/2020, 12:25 - Manish AG: Hello\n03/01/2020, 12:26 - Manish AG: This message was deleted\n03/01/2020, 12:49 - Suyash: You deleted this message\n03/01/2020, 13:01 - Manish AG: <Media omitted>\n

We will be starting with counting no. of messages per sender for this, we will need to define a regex pattern which separates messengers name from the text data.

pattern = re.compile('\d+:\d+\s+-\s+([a-zA-Z0-9]+\s?[a-zA-Z0-9]+\s?[a-zA-Z0-9]+\s?):\s+')  
messengers = re.findall(pattern,data)

Here we have grouped the required messenger name using the **()** notation of regex. The size of the pattern and symbols within () can be changed according to the name of the contact.

To understand how this regex pattern was designed, basic knowledge of regex symbols is required. For reference: <https://regex101.com/>

No. of messages per user

Counting the number of messages per sender & storing results in a dictionary.

count\_messages={}  
for each in messengers:  
 if each in count\_messages.keys():  
 count\_messages[each]+=1  
 else:  
 count\_messages[each]=1  
count\_messages

Output:

{'Suyash': 387, 'Manish AG': 321}

Now to obtain messages separately we will be performing a split operation over the regex pattern we used earlier to get the sender’s name.

messages\_split = pattern.split(data)  
messages\_split[9:11]

Output:

['Suyash', 'Okay\n18/11/2017, ']

Cleaning Messages & Creating DataFrame

Cleaning of messages is very important to obtain data on which various nlp tasks can be performed in future.

The *messages\_split* variable contains sender name followed by his message as next index. So we append the string from *messages\_split* present at index next to messneger name into a new list. The messages are appended in list userwise. First all messages from one user will be appended then from other.

sep\_msgs=[]  
for each in count\_messages.keys():  
 for msg in range(len(messages\_split)):  
 if each == messages\_split[msg]:  
 sep\_msgs.append(messages\_split[msg+1]) #obtaining the message mentioned after sender along with datesprint(len(sep\_msgs))   
sep\_msgs[7]

Output:

708  
'Waiting\n18/11/2017,'

The output above is the total number of messages exchanged and it is equivalent to the sum of values present in *count\_messages* dictionary.

Also the separated messages that we stored in sep\_msgs variable has *\n[date]* present at the end which is not desired. Since dates start from 0,1,2 or 3, therefore we perform a split over ‘\n0’,’\n1',’\n2' & ‘\n3’ and save the first element of the split which contains our required message.

cleaned\_sep\_msg = []  
for each in sep\_msgs:  
 if '\n0' in each:  
 cleaned\_sep\_msg.append(each.split('\n0'))  
 elif '\n1' in each:  
 cleaned\_sep\_msg.append(each.split('\n1'))  
 elif '\n2' in each:  
 cleaned\_sep\_msg.append(each.split('\n2'))  
 elif '\n3' in each:  
 cleaned\_sep\_msg.append(each.split('\n3'))my\_msg = []  
for each in cleaned\_sep\_msg:  
 my\_msg.append(each[0])print(len(my\_msg))  
my\_msg[7]

Output:

707  
'Waiting'

Finally we have stored our messages in *my\_msg* variable, but here we notice that the number of messages have now decreased by 1. This is because the last message exchanged does not have a *\n[date]* followed at its tail due to which when split was being performed, it was not considered. So we can separately append that message at the end of *my\_msg* variable.

Since the*messages\_split*variable contains sender name followed by his message as next index. We will check the 2nd last index and append the leftout message in *my\_msg*variable at appropriate position.

for each in count\_messages.keys():  
 if messages\_split[-2] == each:  
 my\_msg.insert(count\_messages[each]-1,messages\_split[-1])  
len(my\_msg)

Output:

708

Thus we get all our messages captured.

Creating DataFrame

Now in the following code we will be creating a Dataframe with messengers as the column names and messages as the values.

who\_sent\_what = []  
prev = 0  
for each in count\_messages.keys():  
 num = count\_messages[each]  
   
 nex = num+prev  
 messages = my\_msg[prev:nex]  
 who\_sent\_what.append(messages)  
 prev = nex  
who\_sent\_what  
my\_df=pd.DataFrame(who\_sent\_what)  
my\_df = my\_df.transpose()  
my\_df.columns = [list(count\_messages.keys())[0],list(count\_messages.keys())[1]]

In the above code we separate a list of messages for each messenger using keys and their values. The ‘*count\_messages[each]*’ contains number of meassages from a user. We perform slicing on *my\_msg* variable using this info, the variable *who\_sent\_what* will be a list conatining 2 lists, once per user.

This list is later used to create DataFrame.

Gaining Insights

Messages Deleted

We can view who deleted how many messages. Here we need to use ‘*You deleted this message*’ & ‘*This message was deleted*’ explicitly since the messages deleted by sender & receiver are not depicted similarly.

suyash\_deleted = my\_df['Suyash'].value\_counts()['You deleted this message']  
manish\_deleted = my\_df['Manish AG'].value\_counts()['This message was deleted']  
print('Suyash deleted '+ str(suyash\_deleted) + ' messages')  
print('Manish deleted '+ str(manish\_deleted) +' messages')

Output:

Suyash deleted 8 messages  
Manish deleted 4 messages

Emoji Count

We can also see which emoji was used most by which user by using a *emoji* library and defining a function which extracts all the emojis userwise from the messages. The function we will use is defined below. It will parse every string present in the dataframe columnwise and return a list of emojis.

def extract\_emojis(columnname):  
 emojis=[]  
 for string in my\_df[columnname]:  
 my\_str = str(string)  
 for each in my\_str:  
 if each in emoji.UNICODE\_EMOJI:  
 emojis.append(each)  
 return emojis

Creating emojis dataframe userwise and extracting top 5 emojis used by each person.

emoji\_dict={}  
for keys in count\_messages.keys():  
 print(keys)  
 emoji\_dict[keys] = extract\_emojis(keys)  
 emoji\_df = pd.DataFrame(emoji\_dict[keys])  
 print(emoji\_df[0].value\_counts()[:5])

Output:

Suyash  
😂 92  
🏻 13  
😇 10  
🤣 7  
🐒 7  
Name: 0, dtype: int64  
Manish AG  
🤣 28  
😂 12  
😍 10  
😅 8  
🏻 6  
Name: 0, dtype: int64

Who sent more media messages

Since we didn’t include media files while importing the files, therefore they are represented by ‘ *<Media omitted>*’ in our dataframe. Therefore we can view who sent how many media files.

suyash\_media = my\_df['Suyash'].value\_counts()['<Media omitted>']  
manish\_media = my\_df['Manish AG'].value\_counts()['<Media omitted>']print('Suyash sent ' + str(suyash\_media) + ' media files')  
print('Manish sent ' + str(manish\_media) + ' media files')

Output:

Suyash sent 79 media files  
Manish sent 32 media files

Which hour of the day are most messages exchanged?

We will use the same regex pattern used earlier to obtain date from the data and separate hour value by using group method of regex. Matplotlib is used to visualize

hour\_pattern = '(\d+):\d+\s+-\s+\w+\s?\w+?\s?\w+\s?\w+:\s'  
hours = re.findall(hour\_pattern,data)time = pd.DataFrame({'hours':hours})busy\_hours = time['hours'].value\_counts()busy\_hours.sort\_index(inplace=True)  
plt.axes([1,1,1,0.98])  
plt.grid(True)  
busy\_hours.plot.bar()  
plt.xlabel('Hour')  
plt.ylabel('No. of Messages')  
plt.xticks(rotation=0)plt.show()  
%matplotlib inline

Output:

Image for post

