NATURAL LANGUAGE PROCESSING PROJECT REPORT

WhatApp Chat Analyzer

Introduction

We created an app named **whatsapp Chat Analyzer** to analyze the conversation between two persons or a group using **Python** language.

By this ,users can collect the information like total number of messages , words ,media and link shared. On which day, date ,month users are most active and what is the most active hour and who shared most messages, media ,links or emojis . Most common/used words and emojis and by whom it was sent.And to know the emotions of the users while sending the messages we include sentimental analysis.

Dataset Used

Simple whatsapp conversation about whom you want to do analysis.

Implementation

i) ***Preprocessor***

Moving towards the implementation part, firstly we preprocessed the dataset i.e. whatsapp chat by importing the inbuilt libraries pandas and regular expression.

Lets see how this works but before that upload the file in the form of txt.

a-We will first see message extraction using regular expression

```
# for message
pattern2='\d{1,2}/\d{1,2}/\d{2},\s\d{1,2}:\d{2}\s\D{2}\s-\s'
messages = re.split(pattern2, dataset)[1:]
messages
```

```
 'Jaya😊: hello doston\n',
 'Saba: Sab final ho gaya h kya ??\n',
 'Saba: 😂\n',
 'Jaya😊: discuss kre se hoga na\n',
 'Saba: Hii\n',
 'Jaya😊: sab apna idea dete h\n',
 'Saba: Theek h. .\n',
 'Saba: Baaki kaha ho🙃\n',
 'Jaya😊: madam log dawa koe bhejwaega thode\n',
 'Jaya😊: aapko order krna hoga\n',
 'Jaya😊: website bana rh h\n',
 'Jaya😊: ek baar sab aa jao phir discuss krte h\n',
 'Saba: 😂😂\n',
 'Saba: Haa\n',
 'Saba: Sabne msg seen kr liya h😂😂\n',
 'Jaya😊: to reply\n',
```

b-Datetime extraction-: (again by regular expression)

```
# for datetime
pattern2='\d{1,2}/\d{1,2}/\d{2},\s\d{1,2}:\d{2}\s\D{2}\s-\s'
datetime = re.findall(pattern2, dataset)[1:]
datetime
```

```
'1/31/22, 10:24 PM - ',
'1/31/22, 10:27 PM - ',
'1/31/22, 10:27 PM - ',
'1/31/22, 10:37 PM - ',
'1/31/22, 10:39 PM - ',
'1/31/22, 10:39 PM - ',
'1/31/22, 10:39 PM - ',
'1/31/22, 10:39 PM - ',
'1/31/22, 10:46 PM - ',
'1/31/22, 10:47 PM - ',
'1/31/22, 10:48 PM - ',
'1/31/22, 10:48 PM - ',
'1/31/22, 10:48 PM - ',
'1/31/22, 10:48 PM - ',
'1/31/22, 10:48 PM - ',
'1/31/22, 10:48 PM - ',
'1/31/22, 10:48 PM - ',
'1/31/22, 10:49 PM - ',
'1/31/22, 10:50 PM - ',
'1/31/22, 10:50 PM - ',
```

c-Creating a data frame (by using pandas based df)

```
users = []
messages = []
for message in df['user_message']:
    entry = re.split('([\w\W]+?):\s', message)
    if entry[1:]:  # user name
        users.append(entry[1])
        messages.append(" ".join(entry[2:]))
    else:
        users.append('group_notification')
        messages.append(entry[0])
df['user'] = users
df['message'] = messages
df.drop(columns=['user_message'], inplace=True)
df.head()
```

	date	user	message
0	(1/26/22, 7:44 , PM)	group_notification	Messages and calls are end-to-end encrypted. N...
1	(1/26/22, 7:44 , PM)	group_notification	You created group "Diva Power - Hackathon ✌ "\n
2	(1/26/22, 7:45 , PM)	group_notification	Shreya added +91 78008 46759\n
3	(1/26/22, 7:47 , PM)	+91 78008 46759	register krne me koe prblm aa rh h ?\n
4	(1/26/22, 7:49 , PM)	Saba	https://esummit.ecellnitb.com/hackathon.html?u...

d-Extraction of emoji, links , and time, day , month , most used word and most active user-:

```
[ ] num_words = 0                          # num_words -> total number of words
    for msg in df['messages']:
      num_words = num_words + len(msg.split())
    print(num_words)

[ ] # for total number of media shared
    num_media = df[df['messages'] == '<Media omitted>\n'].shape[0]
    print(num_media)

    # for total links shared
    #urls = re.findall('(?:(?:https?|ftp):\/\/)?[\w/\-?=%.]+\.[\w/\-&?=%.]+',dataset)
    pattern2= '\\b((?:https?|ftp|file)://[-a-zA-Z0-9+&@#/%?=~|!:, .;]*[-a-zA-Z0-9+&@#/%=~|])'
    urls = re.findall(pattern2,dataset)
    # print(urls)
    print("Num_links=",len(urls))
    urls

[ ] # for MOS ACTIVE USER
    x = df['sender'].value_counts().head()
    x

[ ] # ploting graph
    import matplotlib.pyplot as plt
    name=x.index
    count = x.values
    plt.bar(name,count)
    plt.xticks(rotation='vertical')
    plt.show()

[ ] # message_percentages of users
    y = round((df['sender'].value_counts()/df.shape[0])*100,2).reset_index().rename(columns={'index':'User Name'
    y
```

ii) ***App Framework***

After preprocessing , we need to a frame to upload the whatsapp chat using **Streamlit** open source framework for python and create a sidebar where user can upload their conversation chat file(group chats or individual chats).If user will the file then user will get one dropbox below sidebar to select whether the user wants to analyze the overall conversation or for a particular user by clicking onto the overall analysis or name of the user.

```
st.sidebar.title("Whatsapp Chat Analyzer")

uploaded_file = st.sidebar.file_uploader("Select a file")                        # for uploading chats for analysis
if uploaded_file is not None:
    bytes_data = uploaded_file.getvalue()
    dataset = bytes_data.decode("utf-8")                                          # converting the data into string
    df = preprocessor.preprocess(dataset)

    # st.dataframe(df)                                                              # to display the dataset

    # fetch unique users
    list_of_users = df['sender'].unique().tolist()                                # list of the users in the chat
    list_of_users.remove('group_notification')
    list_of_users.sort()                                                          # user's names in ascending order
    list_of_users.insert(0,"Overall analysis")                                    # for overall analysis of the group
    selected_sender = st.sidebar.selectbox("Show analysis wrt",list_of_users)     # drop down box for the list of users
```

iii) ***Features***

For total numbers of messages , words ,links and media (individual / overall)

```
def fetch_stats(selected_sender,df):
    if selected_sender != "Overall analysis":                                     # analysis for a single user
        df =  df[df['sender'] == selected_sender]
    # overall analysis of the group
    total_msg = df.shape[0]                                                       # total number of messages
    num_words = 0                                                                 # num_words -> total number of words
    for msg in df['messages']:
        num_words = num_words + len(msg.split())
    num_media = df[df['messages'] == '<Media omitted>\n'].shape[0]                # number of media messages used
    lk =[]
    for msg in df['messages']:
        lk.extend(extract.find_urls(msg))
    num_links = len(lk)                                                           # number of links shared
    return total_msg, num_words,num_media,num_links
```

Most Active User

```
def most_active_users(df):
    x = df['sender'].value_counts().head()
    y = round((df['sender'].value_counts() / df.shape[0]) * 100, 2).reset_index().rename(
        columns={'index': 'User Name', 'sender': 'Percentage'})
    return x,y
```

Word cloud for most used word (using word cloud library)

```
def create_wordcloud(selected_sender,df):
    f = open('stop_words.txt', 'r')
    stop_words = f.read()
    if selected_sender != "Overall analysis":
        df = df[df['sender'] == selected_sender]
    temp = df[df['sender'] != 'group_notification']
    temp = temp[temp['messages'] != '<Media omitted>\n']
    def remove_sw(messages):
        y1=[]
        for word in messages.lower().split():
            if word not in stop_words:
                y1.append(word)
        return " ".join(y1)
    wc = WordCloud(width=500,height=500,min_font_size=10,background_color='white')
    temp['messages'] = temp['messages'].apply(remove_sw)
    df_wc = wc.generate(temp['messages'].str.cat(sep=" "))
    return df_wc
```

Most Used Emojis

For this feature , we installed the emoji library and import library.

```
def mu_emojis(selected_sender,df):
    if selected_sender != 'Overall analysis':
        df = df[df['sender'] == selected_sender]
    emojis= []
    for messages in df['messages']:
        emojis.extend([c for c in messages if c in emoji.UNICODE_EMOJI['en']])
    emojis_mu = pd.DataFrame(Counter(emojis).most_common(len(Counter(emojis))))
    return emojis_mu
```

Monthly and daily Analysis

```
def monthly_ana(selected_sender,df):
    if selected_sender != 'Overall analysis':
        df = df[df['sender'] == selected_sender]
    timeline = df.groupby(['year','month_num','month']).count()['messages'].reset_index()
    time=[]
    for i in range(timeline.shape[0]):
        time.append(timeline['month'][i] + "-" + str(timeline['year'][i]))
    timeline['time']= time
    return timeline

def daily_ana(selected_sender,df):
    if selected_sender != 'Overall analysis':
        df = df[df['sender'] == selected_sender]
    daily_timeline = df.groupby('only_date').count()['messages'].reset_index()
    return daily_timeline
```

Most Active day and month

```
def active_day(selected_sender,df):
    if selected_sender != 'Overall analysis':
        df = df[df['sender'] == selected_sender]
    return df['day_name'].value_counts()

def active_month(selected_sender,df):
    if selected_sender != 'Overall analysis':
        df = df[df['sender'] == selected_sender]
    return df['month'].value_counts()
```

Active Hours **(**for graph we used heatmap**)**

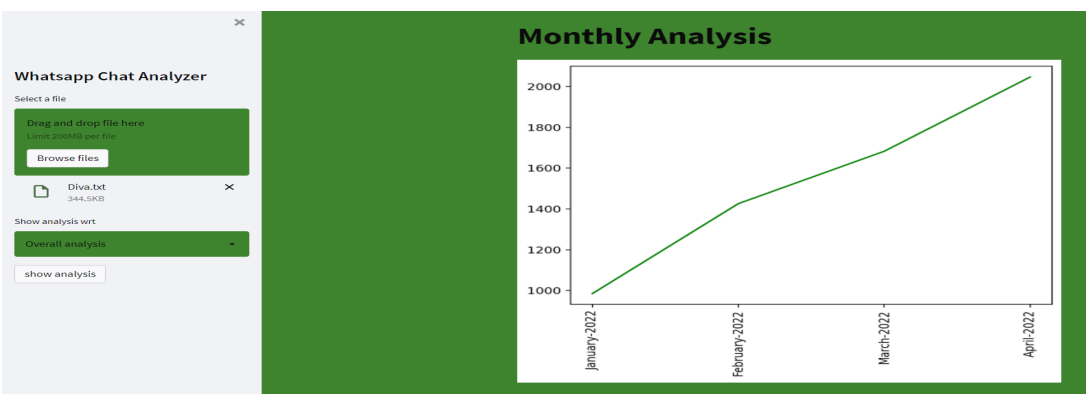
```
def active_hours(selected_sender,df):
    if selected_sender != 'Overall analysis':
        df = df[df['sender'] == selected_sender]
    user_active_hours = df.pivot_table(index='day_name',columns='period',values='messages',aggfunc='count').fillna(0)
    return user_active_hours
```

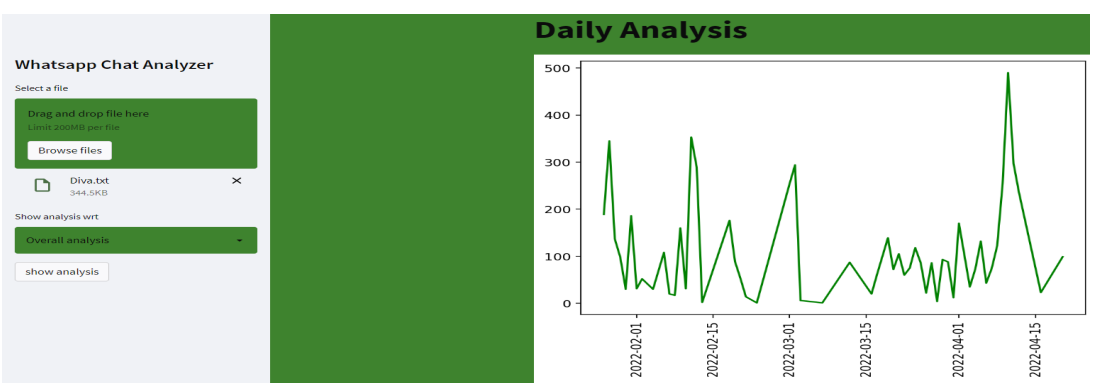
Sentimental Analysis **(we installed nltk toolkit and from nltk we downloaded lexicon-vader)**

```
# sentimental analysis
from nltk.sentiment.vader import SentimentIntensityAnalyzer
sentiments = SentimentIntensityAnalyzer()                    # for each message we are checking its sentiment
st.title('Sentimental Analysis')
df["positive"] = [sentiments.polarity_scores(i)["pos"] for i in df["messages"]]
df["negative"] = [sentiments.polarity_scores(i)["neg"] for i in df["messages"]]
df["neutral"] = [sentiments.polarity_scores(i)["neu"] for i in df["messages"]]
st.dataframe(df)
```

Output

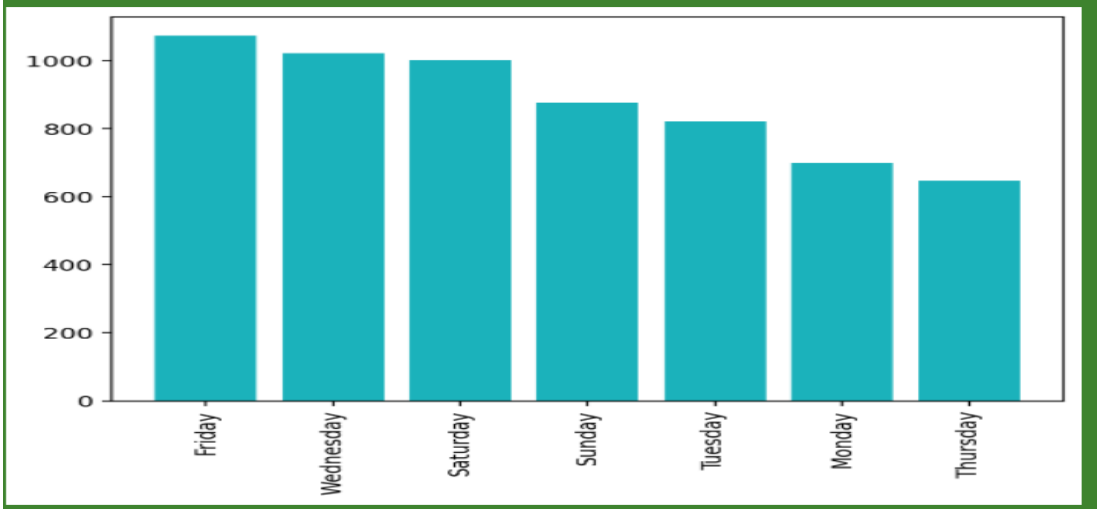
For output we uploaded one chat named diva.txt (group analysis)





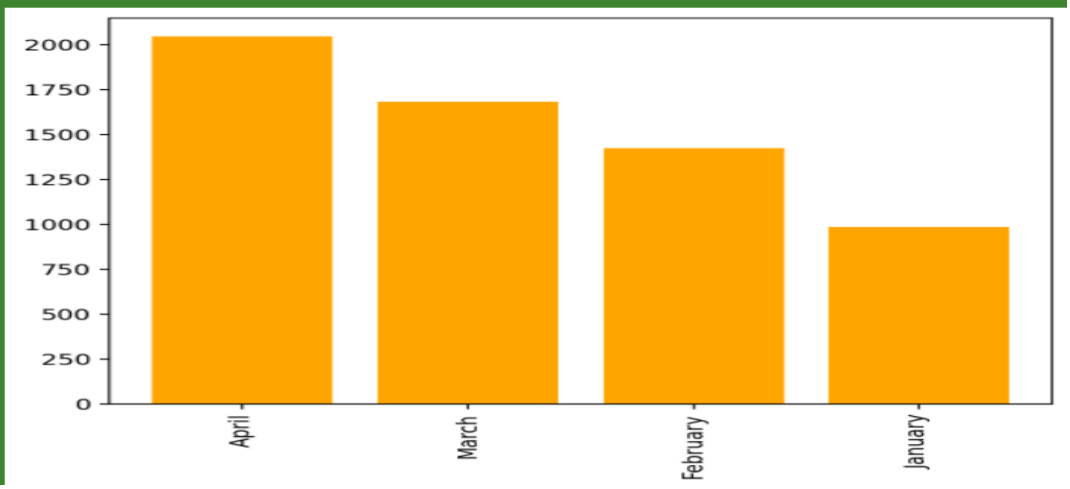
Day Activity

Most Busy Day

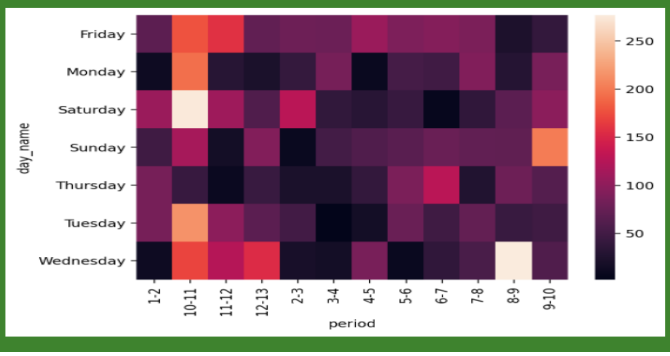


Monthly Activity

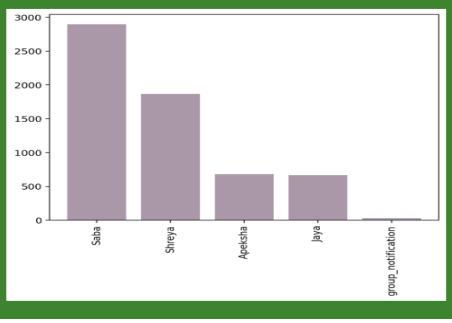
Busiest Month



Active Hours

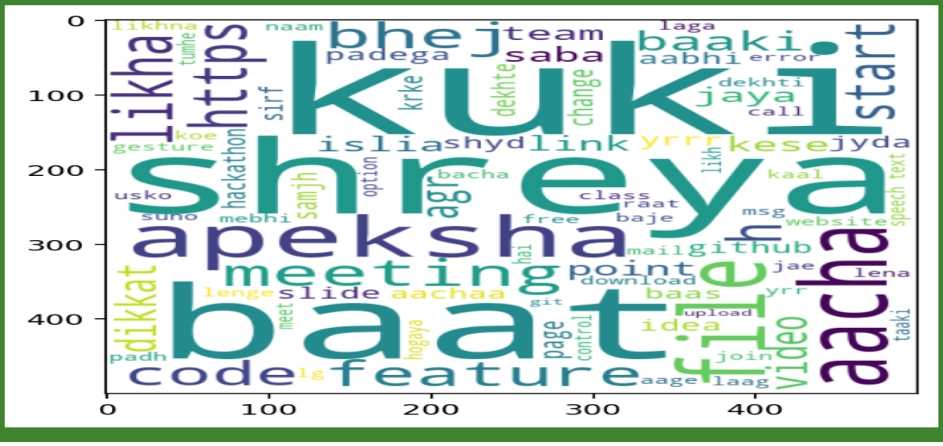


Most Active Users

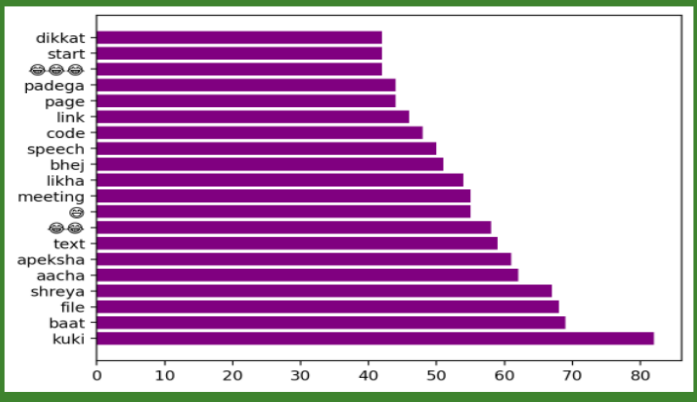


	User Name	Percentage
0	Saba	47.1900
1	Shreya	30.4200
2	Apeksha	11.0600
3	Jaya	10.8500
4	group_notification	0.4900

Most Used Words

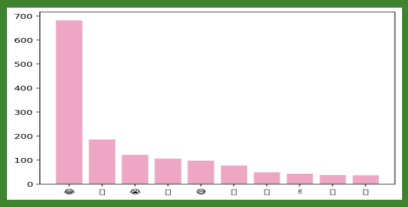


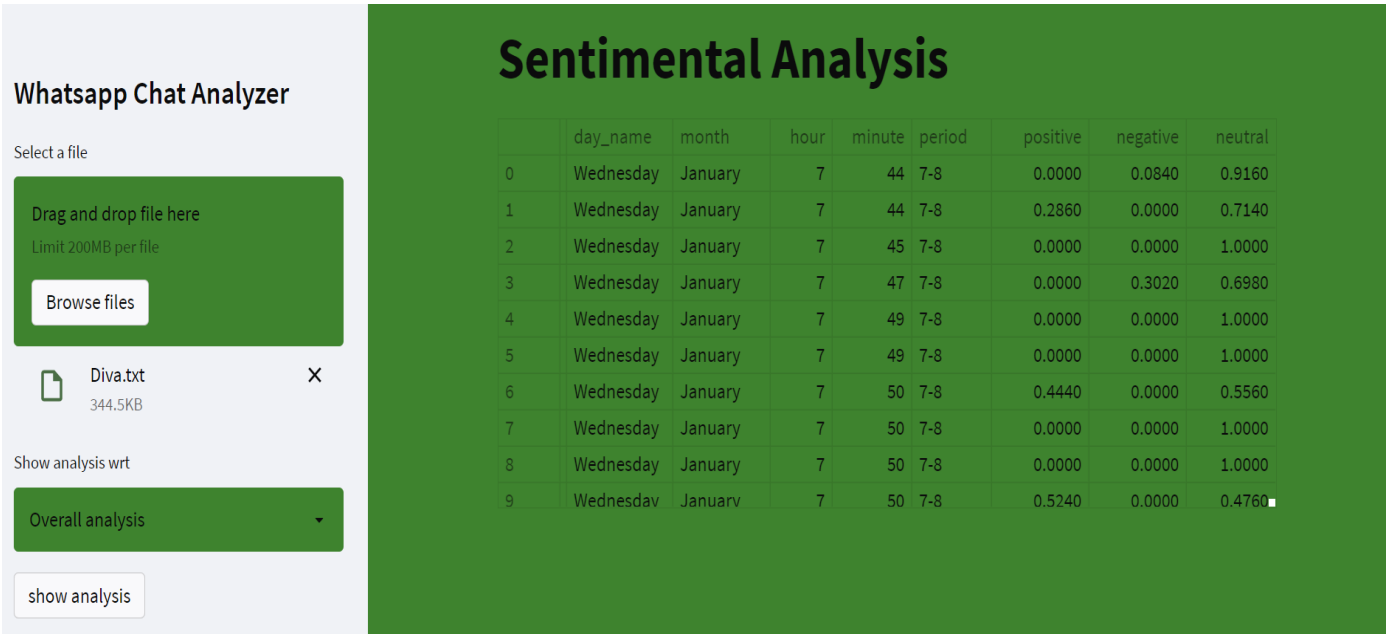
Statistics for Most Used Words



Most Used Emojis

	0	1
0	😂	682
1	🤣	186
2	😭	122
3	⊠	106
4	😊	97
5	😍	77
6	😅	49
7	✌	43
8	😉	37
9	□	36





	day_name	month	hour	minute	period	positive	negative	neutral
0	Wednesday	January	7	44	7-8	0.0000	0.0840	0.9160
1	Wednesday	January	7	44	7-8	0.2860	0.0000	0.7140
2	Wednesday	January	7	45	7-8	0.0000	0.0000	1.0000
3	Wednesday	January	7	47	7-8	0.0000	0.3020	0.6980
4	Wednesday	January	7	49	7-8	0.0000	0.0000	1.0000
5	Wednesday	January	7	49	7-8	0.0000	0.0000	1.0000
6	Wednesday	January	7	50	7-8	0.4440	0.0000	0.5560
7	Wednesday	January	7	50	7-8	0.0000	0.0000	1.0000
8	Wednesday	January	7	50	7-8	0.0000	0.0000	1.0000
9	Wednesday	January	7	50	7-8	0.5240	0.0000	0.4760

Conclusion

By applying the steps/codes mentioned in the preprocessor , framework and features we got the output for analysis of whatsapp chat.

Observations

From the outputs we observed some **challenges** in following features:

1) **Most Used Words**
   For finding the most used words ,firstly we removed the stopwords from the messages (by creating a text file in which both hindi and english stop words were present and remove them from messages) then start the analysis but the analysis was not accurate because in hindi there is no fix spellings for a particular word , so if we include a hindi stop words (which are usually used in hindi chats) and someone is using different alphabets for the same stop words ,then it will not considered that words as stopwords and it will reflect into the Most used words analysis.
2) **Sentimental Analysis :**
   In sentimental analysis ,we observed that, for a particular word the output is coming out as positive sentiment in most of the messages while the output should be negative sentiment.

Made by : Shreya Dalmia (BT19ECE049) and Saba Bano (BT19ECE070)