Whatsapp_Chat_Analysis Part:

Required Libraries:

```
!pip install gradio
!pip install regex
!pip install emoji
!pip install plotly
```

```
Collecting gradio
  Downloading gradio-5.4.0-py3-none-any.whl.metadata (16 kB)
Collecting aiofiles<24.0,>=22.0 (from gradio)
  Downloading aiofiles-23.2.1-py3-none-any.whl.metadata (9.7 kB)
Requirement already satisfied: anyio<5.0,>=3.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (3.7.1)
Collecting fastapi<1.0,>=0.115.2 (from gradio)
  Downloading fastapi-0.115.4-py3-none-any.whl.metadata (27 kB)
Collecting ffmpy (from gradio)
  Downloading ffmpy-0.4.0-py3-none-any.whl.metadata (2.9 kB)
Collecting gradio-client==1.4.2 (from gradio)
  Downloading gradio_client-1.4.2-py3-none-any.whl.metadata (7.1 kB)
Requirement already satisfied: httpx>=0.24.1 in /usr/local/lib/python3.10/dist-packages (from gradio) (0.27.2)
Collecting huggingface-hub>=0.25.1 (from gradio)
  Downloading huggingface_hub-0.26.2-py3-none-any.whl.metadata (13 kB)
Requirement already satisfied: jinja2<4.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (3.1.4)
Collecting markupsafe~=2.0 (from gradio)
  Downloading MarkupSafe-2.1.5-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (3.0 kB)
Requirement already satisfied: numpy<3.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (1.26.4)
Requirement already satisfied: orjson~=3.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (3.10.10)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from gradio) (24.1)
Requirement already satisfied: pandas<3.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (2.2.2)
Requirement already satisfied: pillow<12.0,>=8.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (10.4.0)
Requirement already satisfied: pydantic>=2.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (2.9.2)
Collecting pydub (from gradio)
  Downloading pydub-0.25.1-py2.py3-none-any.whl.metadata (1.4 kB)
Collecting python-multipart==0.0.12 (from gradio)
  Downloading python_multipart-0.0.12-py3-none-any.whl.metadata (1.9 kB)
Requirement already satisfied: pyyaml<7.0,>=5.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (6.0.2)
Collecting ruff>=0.2.2 (from gradio)
  Downloading ruff-0.7.2-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (25 kB)
Collecting safehttpx<1.0,>=0.1.1 (from gradio)
  Downloading safehttpx-0.1.1-py3-none-any.whl.metadata (4.1 kB)
Collecting semantic-version~=2.0 (from gradio)
  Downloading semantic_version-2.10.0-py2.py3-none-any.whl.metadata (9.7 kB)
Collecting starlette<1.0,>=0.40.0 (from gradio)
  Downloading starlette-0.41.2-py3-none-any.whl.metadata (6.0 kB)
Collecting tomlkit==0.12.0 (from gradio)
  Downloading tomlkit-0.12.0-py3-none-any.whl.metadata (2.7 kB)
Requirement already satisfied: typer<1.0,>=0.12 in /usr/local/lib/python3.10/dist-packages (from gradio) (0.12.5)
Requirement already satisfied: typing-extensions~=4.0 in /usr/local/lib/python3.10/dist-packages (from gradio) (4.12.2)
Collecting uvicorn>=0.14.0 (from gradio)
  Downloading uvicorn-0.32.0-py3-none-any.whl.metadata (6.6 kB)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from gradio-client==1.4.2->gradio) (2024.10.0)
Collecting websockets<13.0,>=10.0 (from gradio-client==1.4.2->gradio)
  Downloading websockets-12.0-cp310-cp310-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metad
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.10/dist-packages (from anyio<5.0,>=3.0->gradio) (3.10)
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.10/dist-packages (from anyio<5.0,>=3.0->gradio) (1.3.1)
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-packages (from anyio<5.0,>=3.0->gradio) (1.2.2)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from httpx>=0.24.1->gradio) (2024.8.30)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.10/dist-packages (from httpx>=0.24.1->gradio) (1.0.6)
Requirement already satisfied: h11<0.15,>=0.13 in /usr/local/lib/python3.10/dist-packages (from httpcore==1.*->httpx>=0.24.1->gradio)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.25.1->gradio) (3.16.1)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.25.1->gradio) (2.32.3)
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.25.1->gradio) (4.66.6
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas<3.0,>=1.0->gradio) (2.8
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas<3.0,>=1.0->gradio) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas<3.0,>=1.0->gradio) (2024.2)
```

```
import re
import regex
import pandas as pd
import numpy as np
from collections import Counter
import matplotlib.pyplot as plt
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
import emoji
import plotly.express as px
import gradio as gr
```

```
from textblob import TextBlob
```

Validates date and time format:

```
def date_time(s):
    pattern = '^([0-9]+)(\/)([0-9]+)(\/)([0-9]+), ([0-9]+):([0-9]+)[ ]?(AM|PM|am|pm)? -'
    result = regex.match(pattern, s)
    if result:
        return True
    return False
# Checks for author presence
def find_author(s):
    s = s.split(":")
    if len(s)==2:
        return True
    else:
        return False
   # Extracts date, time, author, and message.
def getDatapoint(line):
    splitline = line.split(' - ')
    dateTime = splitline[0]
    date, time = dateTime.split(", ")
    message = " ".join(splitline[1:])
    if find_author(message):
        splitmessage = message.split(": ")
        author = splitmessage[0]
        message = " ".join(splitmessage[1:])
    else:
        author= None
    return date, time, author, message
```

Extracts and organizes chat messages from a file into a structured format:

```
data = []
conversation = '/content/Whatsapp_chat.txt'
with open(conversation, encoding="utf-8") as fp:
    fp.readline()
    messageBuffer = []
    date, time, author = None, None, None
    while True:
        line = fp.readline()
        if not line:
            break
        line = line.strip()
        if date_time(line):
            if len(messageBuffer) > 0:
                data.append([date, time, author, ' '.join(messageBuffer)])
            messageBuffer.clear()
            date, time, author, message = getDatapoint(line)
            messageBuffer.append(message)
        else:
            messageBuffer.append(line)
```

Creates a DataFrame from chat data, converts dates, and displays summary information:

```
df = pd.DataFrame(data, columns=["Date", 'Time', 'Author', 'Message'])
df['Date'] = pd.to_datetime(df['Date'])
print(df.tail(20))
print(df.info())
print(df.Author.unique())
```

```
            Date     Time                          Author  \
13633 2020-10-02  1:27 am                 +91 73032 50500
13634 2020-10-02  1:46 am       Darshan Rander (TSEC, IT)
13635 2020-10-02  1:47 am                 +91 73032 50500
13636 2020-10-02  1:47 am                 +91 73032 50500
13637 2020-10-02  1:47 am                 +91 73032 50500
13638 2020-10-02  1:47 am                 +91 73032 50500
13639 2020-10-02  1:47 am                 +91 73032 50500
13640 2020-10-02  1:47 am       Darshan Rander (TSEC, IT)
13641 2020-10-02  1:49 am  Shubham Chettiar (TSEC CS, TE)
13642 2020-10-02  1:49 am       Darshan Rander (TSEC, IT)
```

```
13643 2020-10-02  1:50 am  Shubham Chettiar (TSEC CS, TE)
13644 2020-10-02  1:52 am         Tanay Kamath (TSEC, CS)
13645 2020-10-02  1:56 am       Darshan Rander (TSEC, IT)
13646 2020-10-02  1:58 am         Tanay Kamath (TSEC, CS)
13647 2020-10-02  1:58 am         Tanay Kamath (TSEC, CS)
13648 2020-10-02  2:05 am       Darshan Rander (TSEC, IT)
13649 2020-10-02  2:05 am       Darshan Rander (TSEC, IT)
13650 2020-10-02  2:05 am       Darshan Rander (TSEC, IT)
13651 2020-10-02  2:11 am         Tanay Kamath (TSEC, CS)
13652 2020-10-02  2:28 am       Darshan Rander (TSEC, IT)

                                          Message
13633                                <Media omitted>
13634                                        Lucky u
13635                                  Darshan brooo
13636                                Yeh kar na dost
13637                                             ..
13638                                             ..
13639                                    Mil jayegaa
13640                    Okk mene pehle kiya tha firse?
13641  Yeah, many people tried this in the morning an...
13642             Okk firse karte hai...hope so mil jaye
13643                                           Yess☝
13644                                            Amen
13645                Kiya lets see aate hai kya credits
13646                                            ....
13647              Bhai lab pe 5-10 mins spend kiye na?
13648                                         7mins ig
13649                                  MCQs mark kiya
13650                                  Sign-in kiya😂😅
13651                               Incognito se na?
13652                                            Yup
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13653 entries, 0 to 13652
Data columns (total 4 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   Date     13653 non-null  datetime64[ns]
 1   Time     13653 non-null  object
 2   Author   12576 non-null  object
 3   Message  13653 non-null  object
dtypes: datetime64[ns](1), object(3)
memory usage: 426.8+ KB
None
[None '+91 96536 93868' 'Dheeraj Lalwani (TSEC, CS)' '+91 99201 75875'
```

Counts and prints the total number of messages in the DataFrame:

```
total_messages = df.shape[0]
print(total_messages)
```

```
13653
```

Counts and prints the number of media messages in the DataFrame:

```
media_messages = df[df["Message"]=='<Media omitted>'].shape[0]
print(media_messages)
```

```
687
```

Counts and summarizes total messages, media shared, and links in the chat data:

```
URLPATTERN = r'(https?://\S+)'
df['urlcount'] = df.Message.apply(lambda x: regex.findall(URLPATTERN, x)).str.len()
links = np.sum(df.urlcount)

print("Chats between all members")
print("Total Messages: ", total_messages)
print("Number of Media Shared: ", media_messages)
print("Number of Links Shared", links)
```

```
Chats between all members
Total Messages:  13653
Number of Media Shared:  687
Number of Links Shared 766
```

Analyzes WhatsApp chat data by calculating message statistics, filtering out media messages, and summarizing user activity, including message counts, average words, emojis, and links sent:

```
# Read the data from a text file
file_path = '/content/Whatsapp_chat.txt'

# Load the data into a DataFrame
data = []
with open(file_path, 'r') as file:
    for line in file:
        parts = line.strip().split(': ', 1)  # Split into Author and Message
        if len(parts) == 2:
            author, message = parts
            data.append({'Author': author, 'Message': message})

df = pd.DataFrame(data)

# Filtering out media messages
media_messages_df = df[df['Message'] == '<Media omitted>']
messages_df = df.drop(media_messages_df.index)

# Calculating letter and word counts
messages_df['Letter_Count'] = messages_df['Message'].apply(len)
messages_df['Word_Count'] = messages_df['Message'].apply(lambda s: len(s.split()))
messages_df['MessageCount'] = 1

# Users list (You can modify this as needed)
users = messages_df['Author'].unique()  # Get unique users from the DataFrame

# Analyzing messages for each user
for user in users:
    # Filtering messages for the specific user
    req_df = messages_df[messages_df['Author'] == user]

    # Displaying stats for the user
    print(f'Stats of {user} -')
    print('Messages Sent:', req_df.shape[0])

    if req_df.shape[0] > 0:  # Avoid division by zero
        words_per_message = np.sum(req_df['Word_Count']) / req_df.shape[0]
        print('Average Words per message:', words_per_message)
    else:
        print('Average Words per message: 0')

    # Counting media messages
    media_count = media_messages_df[media_messages_df['Author'] == user].shape[0]
    print('Media Messages Sent:', media_count)

    # Counting emojis and links (assuming emojis and URLs are counted elsewhere)
    req_df['emoji'] = req_df['Message'].str.count(r'[\U0001F600-\U0001F64F]')  # Example emoji regex
    req_df['urlcount'] = req_df['Message'].str.count(r'http[s]?://\S+')  # Example URL regex

    emojis = req_df['emoji'].sum()
    print('Emojis Sent:', emojis)

    links = req_df['urlcount'].sum()
    print('Links Sent:', links)

    print('---')  # Separator for clarity
```

```
Stats of 27/01/2020, 7:31 pm - +91 96536 93868 -
Messages Sent: 1
Average Words per message: 5.0
Media Messages Sent: 1
Emojis Sent: 0
Links Sent: 0
---
Stats of 27/01/2020, 7:31 pm - Dheeraj Lalwani (TSEC, CS) -
Messages Sent: 1
Average Words per message: 1.0
Media Messages Sent: 0
Emojis Sent: 0
Links Sent: 0
---
Stats of 27/01/2020, 7:32 pm - Dheeraj Lalwani (TSEC, CS) -
Messages Sent: 2
```

```
Average Words per message: 5.0
Media Messages Sent: 0
Emojis Sent: 0
Links Sent: 0
---
Stats of 27/01/2020, 7:32 pm - +91 96536 93868 -
Messages Sent: 1
Average Words per message: 1.0
Media Messages Sent: 0
Emojis Sent: 0
Links Sent: 0
---
Stats of 27/01/2020, 7:43 pm - +91 99201 75875 -
Messages Sent: 1
Average Words per message: 6.0
Media Messages Sent: 0
Emojis Sent: 0
Links Sent: 0
---
Stats of 27/01/2020, 7:43 pm - Dheeraj Lalwani (TSEC, CS) -
Messages Sent: 1
Average Words per message: 1.0
Media Messages Sent: 0
Emojis Sent: 0
Links Sent: 0
---
Stats of 27/01/2020, 7:44 pm - Dheeraj Lalwani (TSEC, CS) -
Messages Sent: 1
Average Words per message: 4.0
Media Messages Sent: 0
Emojis Sent: 0
Links Sent: 0
---
Stats of 27/01/2020, 7:59 pm - +91 95949 08570 -
Messages Sent: 1
Average Words per message: 3.0
Media Messages Sent: 0
Emojis Sent: 0
Links Sent: 0
---
Stats of 27/01/2020, 8:37 pm - +91 96536 93868 -
```

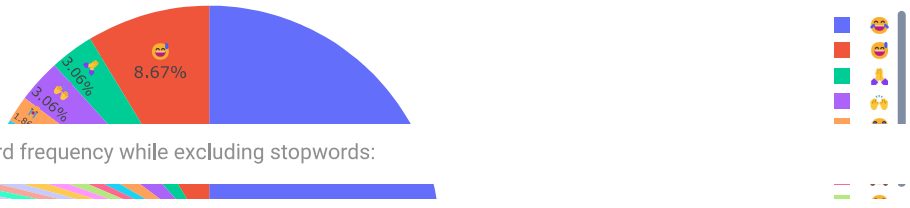
Analyzes WhatsApp chat data to extract and count emoji faces, then visualizes the distribution with a pie chart:

```
# Define the file path to your group chat data
file_path = '/content/Whatsapp_chat.txt'  # Update this to your actual file path

# Load the data
with open(file_path, 'r', encoding='utf-8') as file:
    messages = file.readlines()

# Initialize a list to hold all emoji faces
all_emoji_faces = []

# Define a regex pattern for common emoji faces (modify as needed)
emoji_face_pattern = re.compile(r'[\U0001F600-\U0001F64F]')  # This range covers common emoji faces

# Extract emoji faces from messages
for message in messages:
    # Find all emoji faces in the message
    faces = emoji_face_pattern.findall(message)
    all_emoji_faces.extend(faces)

# Count occurrences of each emoji face
emoji_face_dict = dict(Counter(all_emoji_faces))
emoji_face_dict = sorted(emoji_face_dict.items(), key=lambda x: x[1], reverse=True)

# Print the emoji face counts
for emoji, count in emoji_face_dict:
    print(f"{emoji}: {count}")

# Create a DataFrame for the emoji faces and their counts
emoji_face_df = pd.DataFrame(emoji_face_dict, columns=['emoji', 'count'])

# Create a pie chart of the emoji face counts
fig = px.pie(emoji_face_df, values='count', names='emoji', title='Emoji Face Usage in Group Chats')
fig.update_traces(textposition='inside', textinfo='percent+label')
```

```
fig.show()
```

```
😂: 1896
😅: 224
🙏: 79
🙌: 79
🤦: 48
🥲: 28
🙈: 22
😳: 21
😍: 18
😁: 16
😩: 12
😎: 12
😒: 11
🙃: 10
😊: 8
😬: 8
😤: 8
🥹: 6
🤬: 6
😋: 5
😉: 5
😱: 5
😳: 5
😄: 4
😇: 4
😔: 4
😀: 4
😳: 4
😑: 3
🤦: 3
🤦: 3
😒: 3
😌: 2
😫: 2
🤪: 2
😳: 2
😖: 2
😏: 2
😑: 1
😞: 1
😵: 1
😳: 1
🤦: 1
😈: 1
😳: 1
🥵: 1
🤯: 1
```

Emoji Face Usage in Group Chats



Creates a word cloud from all messages, visualizing word frequency while excluding stopwords:

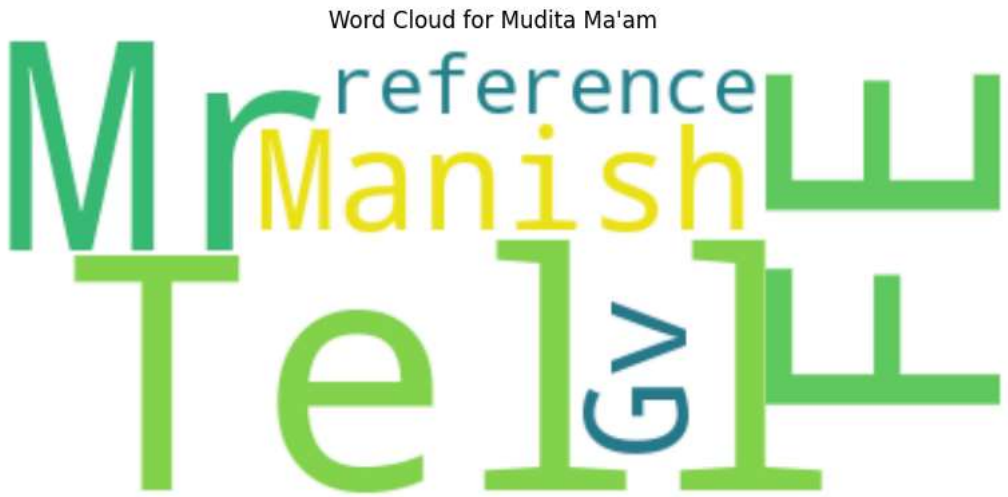
```
text = " ".join(review for review in messages_df.Message)
print ("There are {} words in all the messages.".format(len(text)))
stopwords = set(STOPWORDS)
# Generate a word cloud image
wordcloud = WordCloud(stopwords=stopwords, background_color="white").generate(text)
# Display the generated image:
# the matplotlib way:
plt.figure( figsize=(10,5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

```
There are 390276 words in all the messages.
```

Generates and displays individual word clouds for each author in the WhatsApp chat data, visualizing their unique word usage:

```
# Read the data from a text file
file_path = '/content/Whatsapp_chat.txt'

# Load the data into a DataFrame
data = []
with open(file_path, 'r', encoding='utf-8') as file:
    for line in file:
        # Use regex to capture the timestamp, author, and message
        match = re.match(r'^\[(.*?)] (.*?): (.*)$', line.strip())
        if match:
            date_time, author, message = match.groups()
            data.append({'Author': author, 'Message': message})

# Check if any data was loaded
if not data:
    print("No valid messages found in the file.")
else:
    messages_df = pd.DataFrame(data)

    # Get unique authors from the messages
    authors = messages_df['Author'].unique()

    # Generate word clouds for each author
    for author in authors:
        # Filter messages for the specific author
        dummy_df = messages_df[messages_df['Author'] == author]

        if dummy_df.empty:
            print(f"No messages found for {author}.")
            continue

        # Combine all messages into a single text
        text = " ".join(review for review in dummy_df.Message)

        # Set of stopwords
        stopwords = set(STOPWORDS)

        # Generate a word cloud image
        print('Generating word cloud for:', author)
        wordcloud = WordCloud(stopwords=stopwords, background_color="white").generate(text)

        # Display the generated image
        plt.figure(figsize=(10, 5))
        plt.imshow(wordcloud, interpolation='bilinear')
        plt.axis("off")
        plt.title(f'Word Cloud for {author}')
        plt.show()
```

```
Generating word cloud for: Mudita Ma'am
```



Sentiments_Analysis Part

Required Libraries:

```
import re
import pandas as pd
import numpy as np
from nltk.sentiment import SentimentIntensityAnalyzer
from collections import Counter
import matplotlib.pyplot as plt
from PIL import Image
import nltk
nltk.download('vader_lexicon')

from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
```

Extract Time:

```
def date_time(s):
    pattern = '^([0-9]+)(\/)([0-9]+)(\/)([0-9]+), ([0-9]+):([0-9]+)[ ]?(AM|PM|am|pm)? -'
    result = re.match(pattern, s)
    return bool(result)
```

Find Authors or Contacts:

```
def find_author(s):
    s = s.split(":")
    return len(s) == 2
```

Finding Messages:

```
def getDatapoint(line):
    splitline = line.split(' - ')
    dateTime = splitline[0]
    date, time = dateTime.split(", ")
    message = " ".join(splitline[1:])
    if find_author(message):
        splitmessage = message.split(": ")
        author = splitmessage[0]
        message = " ".join(splitmessage[1:])
    else:
        author = None
    return date, time, author, message
```

Reading and parsing data:

```
data = []
conversation = 'Whatsapp_chat.txt'
with open(conversation, encoding="utf-8") as fp:
    fp.readline()
    messageBuffer = []
    date, time, author = None, None, None
    while True:
        line = fp.readline()
        if not line:
            break
        line = line.strip()
        if date_time(line):
            if messageBuffer:
                data.append([date, time, author, ' '.join(messageBuffer)])
            messageBuffer.clear()
            date, time, author, message = getDatapoint(line)
            messageBuffer.append(message)
        else:
            messageBuffer.append(line)
```

Creating DataFrame:

```
df = pd.DataFrame(data, columns=["Date", 'Time', 'Author', 'Message'])
df['Date'] = pd.to_datetime(df['Date'])
data = df.dropna()
```

```
<ipython-input-21-a7f0c0a2e89a>:2: UserWarning:

Parsing dates in %d/%m/%Y format when dayfirst=False (the default) was specified. Pass `dayfirst=True` or specify a format to silence th
```

Sentiment Analysis:

```
sid = SentimentIntensityAnalyzer()
data["Positive"] = data["Message"].apply(lambda x: sid.polarity_scores(x)["pos"])
data["Negative"] = data["Message"].apply(lambda x: sid.polarity_scores(x)["neg"])
data["Neutral"] = data["Message"].apply(lambda x: sid.polarity_scores(x)["neu"])
```

```
<ipython-input-22-8cf915a4cf1c>:2: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co

<ipython-input-22-8cf915a4cf1c>:3: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co

<ipython-input-22-8cf915a4cf1c>:4: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
```

```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
```

Display Data with Sentiment Scores:

```
print(data[['Date', 'Author', 'Message', 'Positive', 'Negative', 'Neutral']].head(10))
```

```
           Date                     Author  \
112 2020-01-27           +91 96536 93868
113 2020-01-27           +91 96536 93868
114 2020-01-27  Dheeraj Lalwani (TSEC, CS)
115 2020-01-27  Dheeraj Lalwani (TSEC, CS)
116 2020-01-27           +91 96536 93868
117 2020-01-27  Dheeraj Lalwani (TSEC, CS)
118 2020-01-27           +91 99201 75875
119 2020-01-27  Dheeraj Lalwani (TSEC, CS)
120 2020-01-27  Dheeraj Lalwani (TSEC, CS)
121 2020-01-27           +91 95949 08570

                                       Message  Positive  Negative  \
112                                <Media omitted>     0.000     0.000
113                             Give it a try ....     0.000     0.000
114                                      Alright     1.000     0.000
115                         We can make this a trend     0.000     0.000
116                                          Sure     1.000     0.000
117                         Everyday a new challenge     0.394     0.000
118                 More of logic or algo problems^     0.000     0.000
119                                          Yeah     1.000     0.000
120  Something or the other So that we can build ou...     0.149     0.168
121                                     Nice idea 🤘     0.737     0.000

     Neutral
112    1.000
113    1.000
114    0.000
115    1.000
116    0.000
117    0.606
118    1.000
119    0.000
120    0.683
121    0.263
```

Summing and calculating average sentiment scores:

```
x = data["Positive"].mean()
y = data["Negative"].mean()
z = data["Neutral"].mean()
```

Determining Overall Sentiment:

```
def sentiment_score(avg_pos, avg_neg, avg_neu):
    if (avg_pos > avg_neg) and (avg_pos > avg_neu):
        return f"Overall Sentiment: Positive 😊 (Positive: {avg_pos:.2f}, Negative: {avg_neg:.2f}, Neutral: {avg_neu:.2f})"
    elif (avg_neg > avg_pos) and (avg_neg > avg_neu):
        return f"Overall Sentiment: Negative 😡 (Positive: {avg_pos:.2f}, Negative: {avg_neg:.2f}, Neutral: {avg_neu:.2f})"
    else:
        return f"Overall Sentiment: Neutral 😐 (Positive: {avg_pos:.2f}, Negative: {avg_neg:.2f}, Neutral: {avg_neu:.2f})"
```

Print the final sentiment score:

```
print(sentiment_score(x, y, z))
```

```
Overall Sentiment: Neutral 😐 (Positive: 0.14, Negative: 0.04, Neutral: 0.78)
```

```
def analyze_chat(file):
    with open(file.name, 'r', encoding='utf-8') as f:
        chat_data = f.read()

    # Split messages based on line breaks
```

```
    messages = chat_data.split('\n')

    total_messages = len(messages)
    total_links = sum(bool(re.search(r'http[s]?://', message)) for message in messages)
    total_media_files = sum('media' in message.lower() for message in messages)

    # Sentiment analysis
    positive_count = 0
    negative_count = 0
    neutral_count = 0

    for message in messages:
        analysis = TextBlob(message)
        # Classify the message sentiment
        if analysis.sentiment.polarity > 0:
            positive_count += 1
        elif analysis.sentiment.polarity < 0:
            negative_count += 1
        else:
            neutral_count += 1

    return (total_messages, total_links, total_media_files,
            positive_count, negative_count, neutral_count)

# Create a Gradio interface
interface = gr.Interface(
    fn=analyze_chat,
    inputs=gr.File(label="Upload WhatsApp Chat (text file)"),
    outputs=[
        gr.Textbox(label="Total Number of Messages"),
        gr.Textbox(label="Total Links Shared"),
        gr.Textbox(label="Total Media Files Shared"),
        gr.Textbox(label="Total Positive Messages"),
        gr.Textbox(label="Total Negative Messages"),
        gr.Textbox(label="Total Neutral Messages"),
    ],
    title="WhatsApp Chat Analyzer",
    description="Upload a WhatsApp chat text file to analyze the total messages, links shared, media files shared, and sentiment."
)

# Launch the interface
interface.launch()
```

```
Running Gradio in a Colab notebook requires sharing enabled. Automatically setting `share=True` (you can turn this off by setting `share

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://f83cf21e0c48bc3f56.gradio.live

This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working c
```