EXPLORATORY DATA ANALYSIS AND EMOTION DETECTION ON WHATSAPP

By :

Purushottam Kumar

2020178043 MCA (R)

Project Guide

Dr. T. Mala

(Associate Professor)

OBJECTIVE

- Existing system has limitation to analyze text data especially chat conversation and detecting emotion from text is also a major concern.
- To develop a system that can analyze WhatsApp chats using **Exploratory Data Analysis** and illustrate in more visual format to identify hidden insights.
- To detect emotion from text using **LSTM (Long Short Term Memory)**

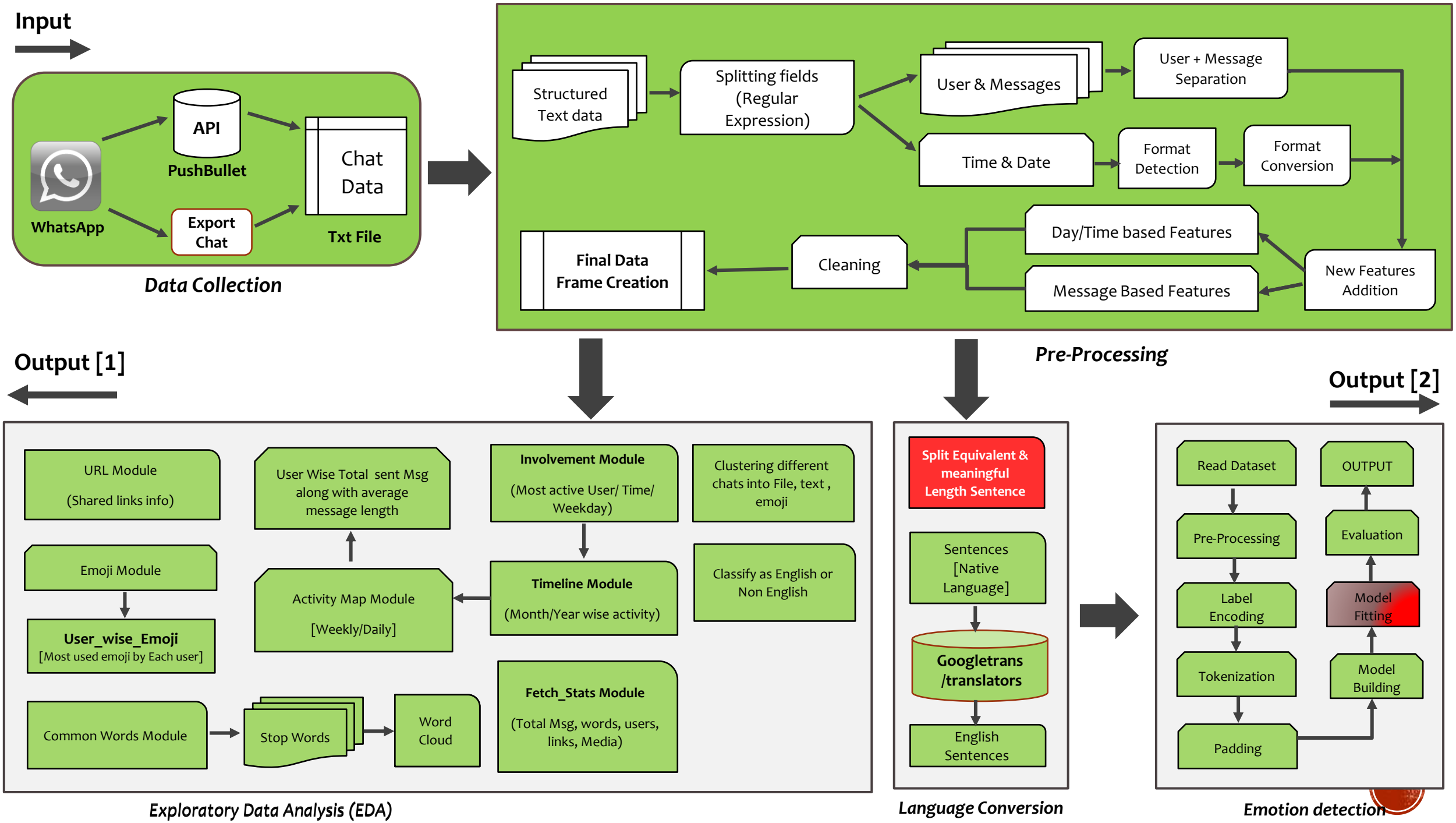
LITERATURE SURVEY

S.No	Title	Problem Statement	Approach	Pros	Cons
1	*Text Classification based Behavioural Analysis of WhatsApp Chats" (June 2020)* **S. Dahiya, Astha Mohta, Atishay Jain**	Classification and analysing sentiment on the basis of emotion and categorise them	Behaviour analysis	To classify emotions (Pos, Neg, Neutral) into six different emotions and uses their Neutrality to weigh them.	Only 72.9% accuracy against a set of pre-classified data.
2	*WhatsApp Network Group Chat Analysis Using Python Programming* **Blessing Nwamaka Iduh** *International Journal of Engineering & Applied Science (IJLTEMAS) Volume IX, Feb 2020*	Analysis of the chats and users in certain groups, to ascertain the level of participation of members in the group chat	Exploratory Analysis	Analysis of the top 10 & top 20 users using the Python libraries	Can be implemented only on (24 hr clock & mm/dd/yyyy) format
3	*Analysing and Predicting the Emotion of WhatsApp Chats Using Sentiment Analysis* **Chinthapanti B Sai Reddy, Kowshik S, Rakesh, Gopichand G**	Using LSTM model to categorize the behavior into Angry, Sad, Fear, Happy, Surprised, Excited	Sentiment Analysis	Using priority based model to classify different emotions	Huge amount of data required for analysis

LITERATURE SURVEY

S.No	Title	Problem Statement	Approach	Pros	Cons
4	*People's Behavior Analysis in Chat Message using Natural Language Processing* V.Selina Annie Retna, P. Brundha, G RajKumar, Mar-2019	To analyze the sentimental flirt analysis in the private setting of a group in the WhatsApp chat to obtain the exact result for the analysis	Data mining, Natural Language Processing	Web Scrapping the WhatsApp chat and sentimental analysis using deep learning model	for storage the cost was little high for using the cloud storage and can be reduced in the future works
5	*Whatsapp Chat Analyzer* Ravishankara K, Dhanush, Vaisakh, Srajan I S *International Journal of Engineering Research & Technology (IJERT)* *Vol. 9 Issue 05, May-2020*	Using python libraries to show level of participation of the various individuals on the given WhatsApp group and addiction	Exploratory data analysis using tools ex-matplotlib, seaborn	Using Data pre-processing & Exploratory data analysis, then apply sentiment analysis algorithm	Limitation of the neglecting of some common words during the chat conversation

ARCHITECTURE DIAGRAM

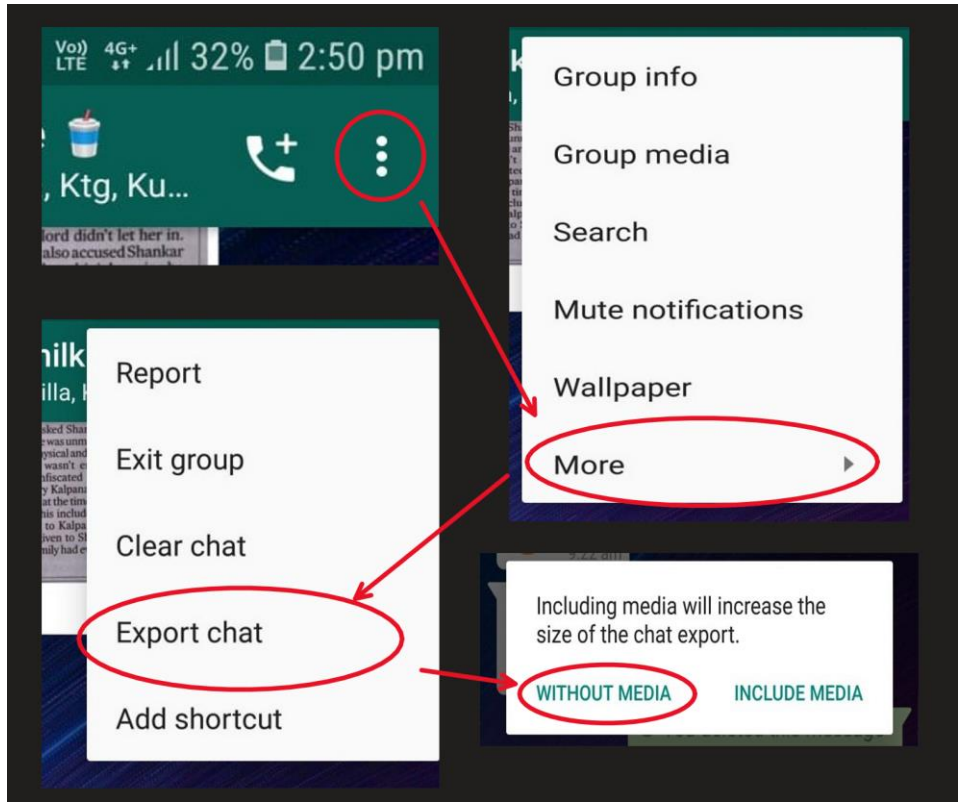

Input
API
PushBullet
WhatsApp
Export Chat
Chat Data
Txt File
Data Collection
Structured Text data
Splitting fields (Regular Expression)
User & Messages
User + Message Separation
Time & Date
Format Detection
Format Conversion
Day/Time based Features
Message Based Features
New Features Addition
Cleaning
Final Data Frame Creation
Pre-Processing
Output [1]
URL Module
(Shared links info)
User Wise Total sent Msg along with average message length
Involvement Module
(Most active User/ Time/ Weekday)
Clustering different chats into File, text , emoji
Emoji Module
User_wise_Emoji
[Most used emoji by Each user]
Activity Map Module
[Weekly/Daily]
Timeline Module
(Month/Year wise activity)
Classify as English or Non English
Common Words Module
Stop Words
Word Cloud
Fetch_Stats Module
(Total Msg, words, users, links, Media)
Exploratory Data Analysis (EDA)
Split Equivalent & meaningful Length Sentence
Sentences [Native Language]
Googletrans /translators
English Sentences
Language Conversion
Output [2]
Read Dataset
Pre-Processing
Label Encoding
Tokenization
Padding
Model Building
Model Fitting
Evaluation
OUTPUT
Emotion detection

MODULES

- Data Collection Module
- Preprocessing Module
- Exploratory Data Analysis Module
- LSTM Module

DATA COLLECTION MODULE-I

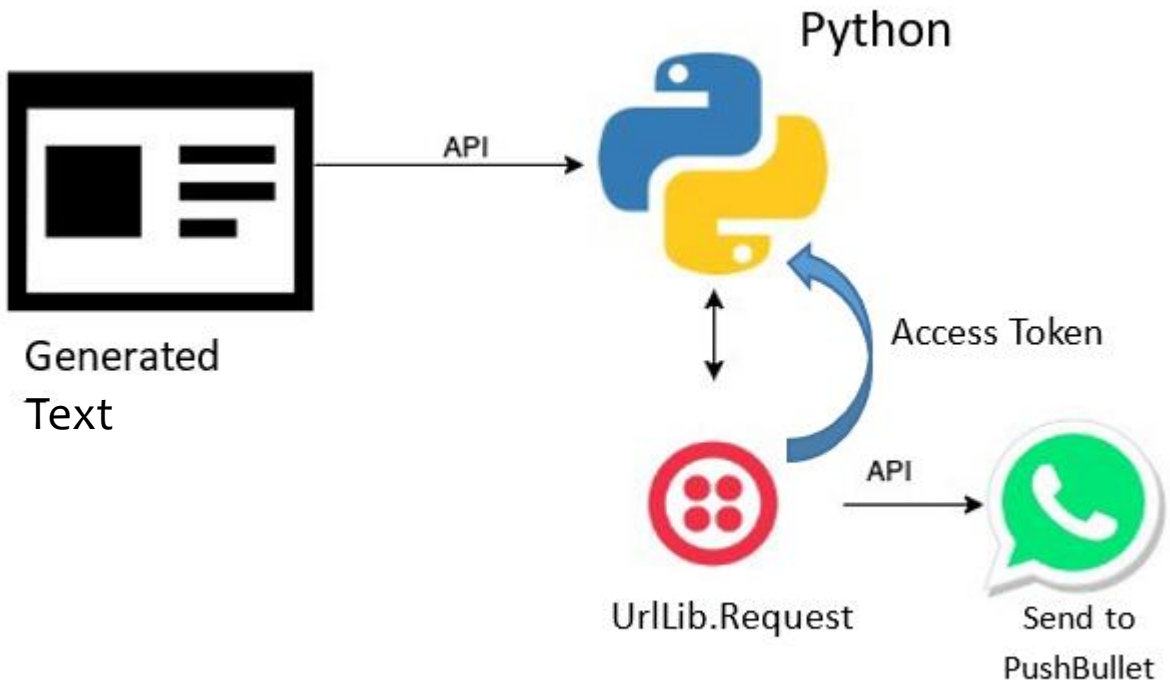
Method – 1 (Via Mobile)



- **STEPS TO Follow**
  - Launch the WhatsApp on phone.
  - Now open the individual or group chat.
  - Then tap on More options or three dots
  - Click More, then tap on Export chat.
  - Choose export without media.

DATA COLLECTION MODULE-II

Method – 2 (Using PushBullet Api)



- **STEPS TO Follow**
  - Download **Pushbullet** app on mobile and sign up with Gmail.
  - Open PushBullet.com in chrome browser and sign in using same Gmail
  - In Mobile Export chat to Pushbullet.
  - Go to settings > account > access Token(API) > Create.
  - pb = Pushbullet(API_KEY)
  - Data = pb.get_pushes()
  - latest = Data[0]
  - url = latest['file_url']
  - file_path = "chat.txt"
  - urllib.request.urlretrieve(url, file_path)

PRE-PROCESSING MODULE-I

- Responsible for cleaning, clustering, feature addition, visualisation and further pre-process the text data.

1. Time-Stamp separation (Splitting Fields)

- Using Regular Expression extracting the Time-stamp from given conversation.

```
#Handling two different Format
pattern = '\d{1,2}/\d{1,2}/\d{2,4},\s\d{1,2}:\d{2}\s[AP]M\s-\s|\d{1,2}/\d{1,2}/\d{2,4},\s\d{1,2}:\d{2}\s-\s|\d{1,2}/\d{1,2}/\d{2,4},\s\d{1,2}:\d{2}\s[ap]m\s-\s'
all_messages = re.split(pattern,data)[1:]
dates = re.findall(pattern,data)
```

+ Code + Markdown

Input

```
27/01/2020, 10:10 pm - +91 96536 93868: This is my approach
27/01/2020, 10:10 pm - +91 75078 05454: <Media omitted>
27/01/2020, 10:11 pm - +91 96536 93868: Great !! Good use of if else ...
27/01/2020, 10:11 pm - +91 75078 05454: Thanks
27/01/2020, 10:12 pm - Darshan Rander (TSEC, IT): This message was deleted
27/01/2020, 10:12 pm - Darshan Rander (TSEC, IT): this is c++
27/01/2020, 10:15 pm - Dheeraj Lalwani (TSEC, CS): Actually great
27/01/2020, 10:15 pm - +91 96536 93868: At our level the only difference between C and C++ is printf and scanf  vala statement
27/01/2020, 10:15 pm - Dheeraj Lalwani (TSEC, CS): Yes true
27/01/2020, 10:15 pm - +91 75078 05454: Yes
27/01/2020, 10:16 pm - Tanay Kamath (TSEC, CS): <Media omitted>
27/01/2020, 10:16 pm - +91 96536 93868: C is only restricted to functions ..While c++has classes and structures vala stuff
27/01/2020, 10:16 pm - Tanay Kamath (TSEC, CS): <Media omitted>
27/01/2020, 10:16 pm - Tanay Kamath (TSEC, CS): Just check my soln out
27/01/2020, 10:16 pm - Dheeraj Lalwani (TSEC, CS): Yup
27/01/2020, 10:16 pm - Tanay Kamath (TSEC, CS): I got a bit confused...
27/01/2020, 10:29 pm - Tanay Kamath (TSEC, CS): I think char b=32 instead
```

Output

```
27/01/2020, 10:10 pm -
27/01/2020, 10:10 pm -
27/01/2020, 10:11 pm -
27/01/2020, 10:11 pm -
27/01/2020, 10:12 pm -
27/01/2020, 10:12 pm -
27/01/2020, 10:15 pm -
27/01/2020, 10:15 pm -
27/01/2020, 10:15 pm -
27/01/2020, 10:15 pm -
27/01/2020, 10:16 pm -
27/01/2020, 10:16 pm -
27/01/2020, 10:16 pm -
27/01/2020, 10:16 pm -
27/01/2020, 10:16 pm -
27/01/2020, 10:16 pm -
27/01/2020, 10:29 pm -
```

```
+91 96536 93868: This is my approach
+91 75078 05454: <Media omitted>
+91 96536 93868: Great !! Good use of if else ...
+91 75078 05454: Thanks
Darshan Rander (TSEC, IT): This message was deleted
Darshan Rander (TSEC, IT): this is c++
Dheeraj Lalwani (TSEC, CS): Actually great
+91 96536 93868: At our level the only difference between C and C++ is printf and scanf
Dheeraj Lalwani (TSEC, CS): Yes true
+91 75078 05454: Yes
Tanay Kamath (TSEC, CS): <Media omitted>
+91 96536 93868: C is only restricted to functions ..While c++has classes and structures
Tanay Kamath (TSEC, CS): <Media omitted>
Tanay Kamath (TSEC, CS): Just check my soln out
Dheeraj Lalwani (TSEC, CS): Yup
Tanay Kamath (TSEC, CS): I got a bit confused...
Tanay Kamath (TSEC, CS): I think char b=32 instead
```

PRE-PROCESSING MODULE-II

2. Data Frame Creation

- From Text file, creation of dataframe will help in further processing.
- Using pandas to create dataframe from separated timestamp.

```
l1 = list(all_messages)
l2 = list(final_date_time)
s1 = pd.Series(l1, name='user_message')
s2 = pd.Series(l2, name='date')
df = pd.concat([s1,s2], axis=1)
```

[20] ✓ 0.1s

Input

```
27/01/2020, 10:10 pm -
27/01/2020, 10:10 pm -
27/01/2020, 10:11 pm -
27/01/2020, 10:11 pm -
27/01/2020, 10:12 pm -
27/01/2020, 10:12 pm -
27/01/2020, 10:15 pm -
27/01/2020, 10:15 pm -
27/01/2020, 10:15 pm -
27/01/2020, 10:15 pm -
27/01/2020, 10:16 pm -
27/01/2020, 10:16 pm -
27/01/2020, 10:16 pm -
27/01/2020, 10:16 pm -
27/01/2020, 10:16 pm -
27/01/2020, 10:16 pm -
27/01/2020, 10:29 pm -
```

+

```
+91 96536 93868: This is my approach
+91 75078 05454: <Media omitted>
+91 96536 93868: Great !! Good use of if else ...
+91 75078 05454: Thanks
Darshan Rander (TSEC, IT): This message was deleted
Darshan Rander (TSEC, IT): this is c++
Dheeraj Lalwani (TSEC, CS): Actually great
+91 96536 93868: At our level the only difference between C and C++ is printf and scanf
Dheeraj Lalwani (TSEC, CS): Yes true
+91 75078 05454: Yes
Tanay Kamath (TSEC, CS): <Media omitted>
+91 96536 93868: C is only restricted to functions ..While c++has classes and structures
Tanay Kamath (TSEC, CS): <Media omitted>
Tanay Kamath (TSEC, CS): Just check my soln out
Dheeraj Lalwani (TSEC, CS): Yup
Tanay Kamath (TSEC, CS): I got a bit confused...
Tanay Kamath (TSEC, CS): I think char b=32 instead
```

Output

user_message	date
Messages and calls are end-to-end encrypted. N...	2021-08-21 18:38:00
Mohana CEG: Hii Kishan\n	2021-08-21 18:38:00
Mohana CEG: I'm Mohana\n	2021-08-21 18:39:00
Mohana CEG: Rohan's MCA class mate\n	2021-08-21 18:39:00
Mohana CEG: I had doubt in Background check fi...	2021-08-21 18:39:00

PRE-PROCESSING MODULE-III

3. Date and Time Format conversation

- Converting 12-hr time to 24-hr clock.
- Detect different date format an convert it to YYYY/MM/DD

4. User and Message Separation

- Using Regular Expression pattern to splitting message and user

```
df['user'] = users
df['message'] = messages
df.drop(columns=['user_message'], inplace=True)
```
[23] ✓ 0.2s

Seperating users and messages

```
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])
```
[195]

Input

user_message	date
Messages and calls are end-to-end encrypted. N...	2021-08-21 18:38:00
Mohana CEG: Hii Kishan\n	2021-08-21 18:38:00
Mohana CEG: I'm Mohana\n	2021-08-21 18:39:00
Mohana CEG: Rohan's MCA class mate\n	2021-08-21 18:39:00
Mohana CEG: I had doubt in Background check fi...	2021-08-21 18:39:00

Output

```
df.sample(10)
```
[26] ✓ 0.4s

	date	user	message
6775	2020-05-28 03:24:00	Kartik Soneji (TSEC, CS)	Oh, I thought you were talking about https://d...
7194	2020-06-06 22:03:00	Dheeraj Lalwani (TSEC, CS)	<Media omitted>\n
13441	2020-09-30 12:40:00	Darshan Rander (TSEC, IT)	Tereko toh pakka aya hoga...GOOGLE😂\n
8459	2020-07-05 13:38:00	Dheeraj Lalwani (TSEC, CS)	let the results come once\n
7780	2020-06-16 14:29:00	Dheeraj Lalwani (TSEC, CS)	Are haa\n
4478	2020-04-11 11:41:00	Darshan Rander (TSEC, IT)	Append your Codeforces handle for Mashup - 11...
11158	2020-09-03 13:01:00	Kartik Soneji (TSEC, CS)	Yes.\n
6678	2020-05-28 02:32:00	Kartik Soneji (TSEC, CS)	ESPECIALLY CSS/JavaScript libraries.\n
8830	2020-07-11 18:52:00	Farhan Irani (TSEC IT, SE)	This message was deleted\n
1065	2020-02-20 22:42:00	Pratik K (TSEC CS, SE)	He's my teammate btw\n

PRE-PROCESSING MODULE-IV

5. Feature Addition / Extraction

- Extracting Day, Month, Year from Date using RegEx.
- Extracting Hour, Min from Time
- Adding them as new feature in dataframe.
- Adding two more features as Word count, total characters involved
- Total attributes reached up to 26

```
df['only_date'] = df['date'].dt.date
df['year'] = df['date'].dt.year
df['month_num'] = df['date'].dt.month
df['month'] = df['date'].dt.month_name()
df['day'] = df['date'].dt.day
df['day_name'] = df['date'].dt.day_name()
df['hour'] = df['date'].dt.hour
df['minute'] = df['date'].dt.minute
```

Converting Given Time to Day part

```
df['day_part'] = (df['hour']% 24 + 4) // 4
df['day_part'].replace({1: 'Late Night',
                        2: 'Early Morning',
                        3: 'Morning',
                        4: 'Noon',
                        5: 'Evening',
                        6: 'Night'}, inplace=True)
```

6. Final Data Frame

```
df.sample(5)
```

✓ 0.2s

	date	user	message	only_date	year	month_num	month	day	day_name	hour	...	total-words	total-characters	total-unique-words	total-urls	Media_Count	total-emojis	total-emojis-no-numbers	pur
13237	2020-09-28 16:23:00	Tanay Kamath (TSEC, CS)	<Media omitted>\n	2020-09-28	2020	9	September	28	Monday	16	...	2	16	2	0	1	0	0	
9574	2020-08-10 17:36:00	+91 99304 97064	+919167007001.vcf (file attached)\n	2020-08-10	2020	8	August	10	Monday	17	...	3	34	3	0	0	0	0	
11665	2020-09-12 22:20:00	Tanay Kamath (TSEC, CS)	by the way C/C++ are very unforgiving\n	2020-09-12	2020	9	September	12	Saturday	22	...	7	38	7	0	0	0	0	
6720	2020-05-28 02:54:00	Kartik Soneji (TSEC, CS)	I try to avoid BootStrap alltogether.\n	2020-05-28	2020	5	May	28	Thursday	2	...	6	38	6	0	0	0	0	
		Darshan																	

EXPLORATORY DATA ANALYSIS-I

- Multiples sub-modules are involved in EDA process.

1. **Fetch_stats sub-module**
   - Going through all the messages and filtering the users, URLs , Message and words and counting them.
   - For URL identification URLSurf library is used.
   - Returning statistical information from function

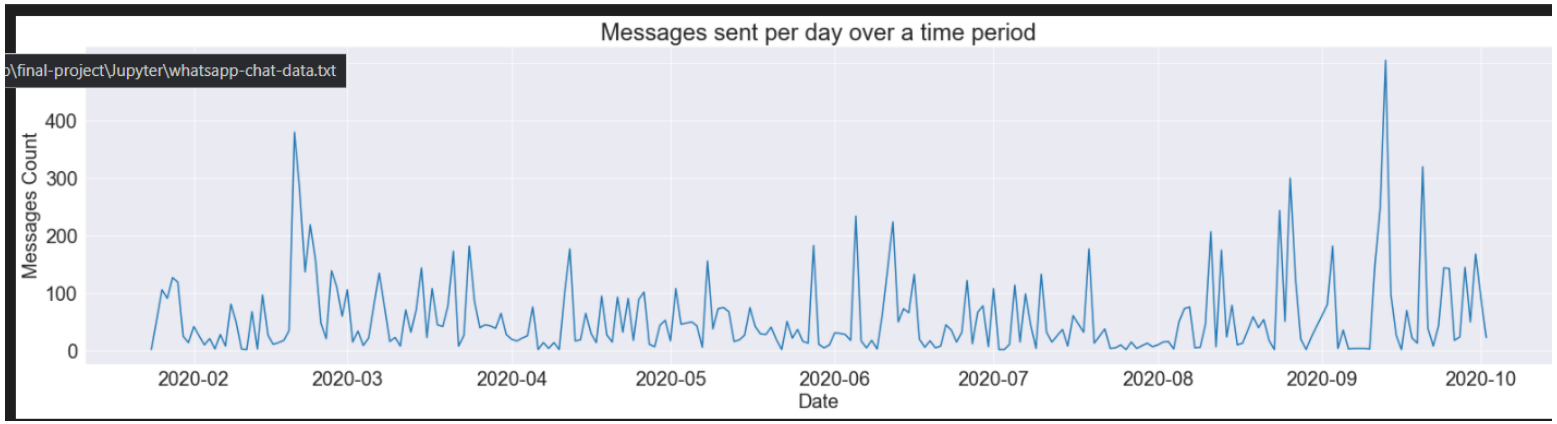
Output

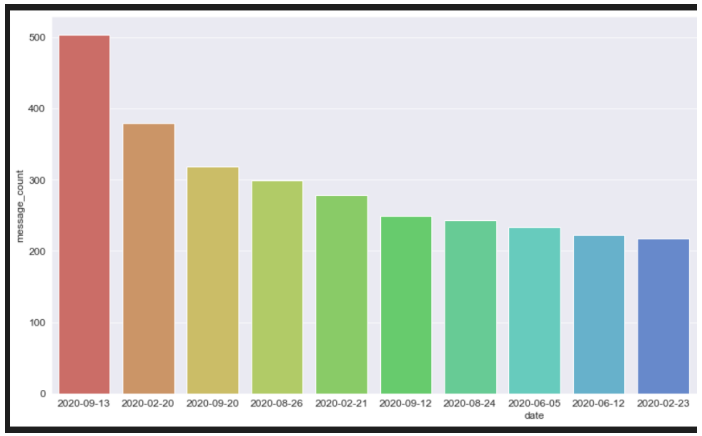
```
num_messages, words, num_media_messages, num_links = fetch_stats('Overall',df)
print('Total MSG : ', num_messages)
print('Total Words : ', words)
print('Total Media : ', num_media_messages)
print('Total Links : ', num_links)
```

[474]

```
Total MSG :  13655
Total Words :  104364
Total Media :  687
Total Links :  1035
```

2. **Timeline sub-module**
   - Analysing the time in a day and weekday during which most conversation has been occurred.
   - It also helps in predicting that which tine us best to send msg so that we can get response



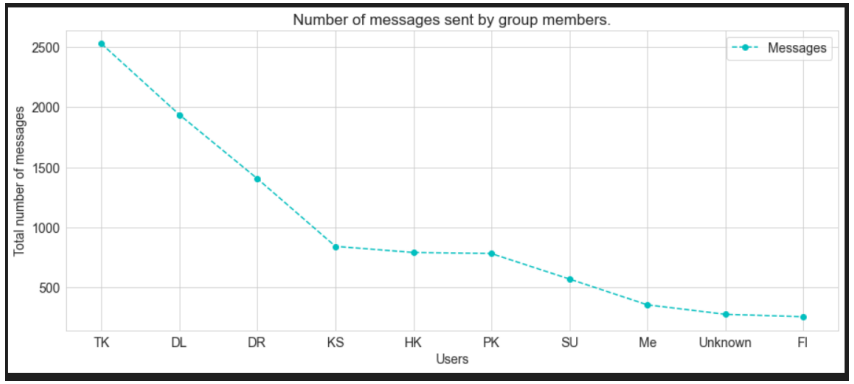


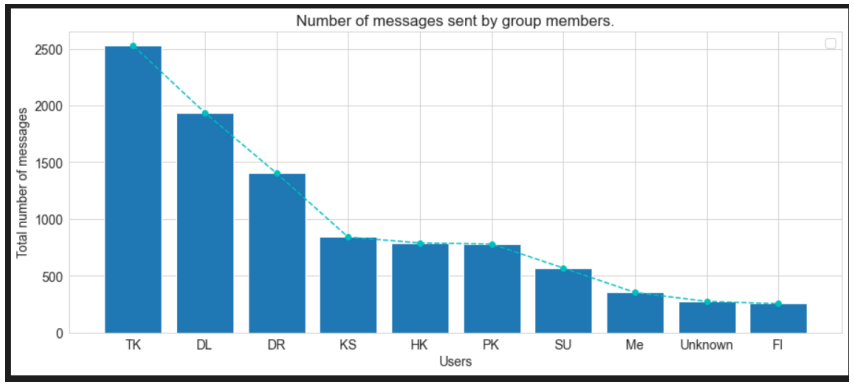
EXPLORATORY DATA ANALYSIS-II

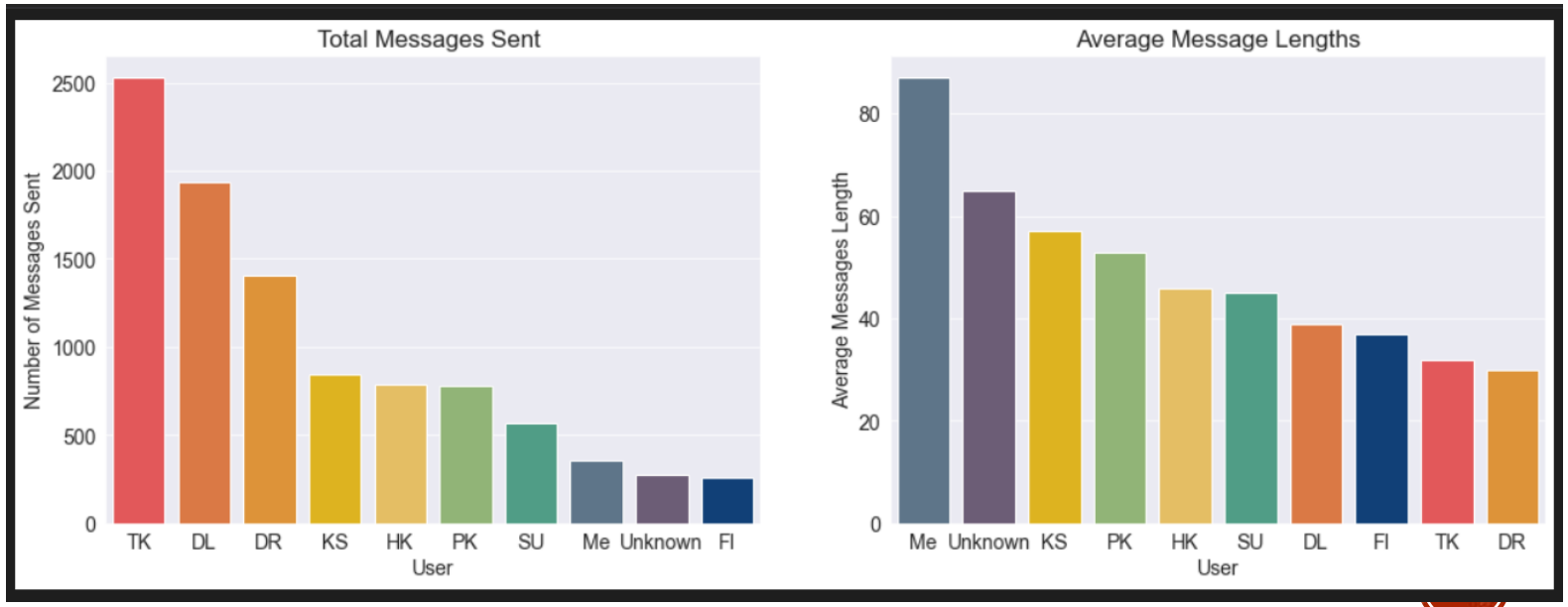
3. Total Msg and avg word length sub-module

- Separating each user message and applying count to identify busiest user.
- Fetching top users from chat by most no of messages and average length of messages.

Output







EXPLORATORY DATA ANALYSIS-III

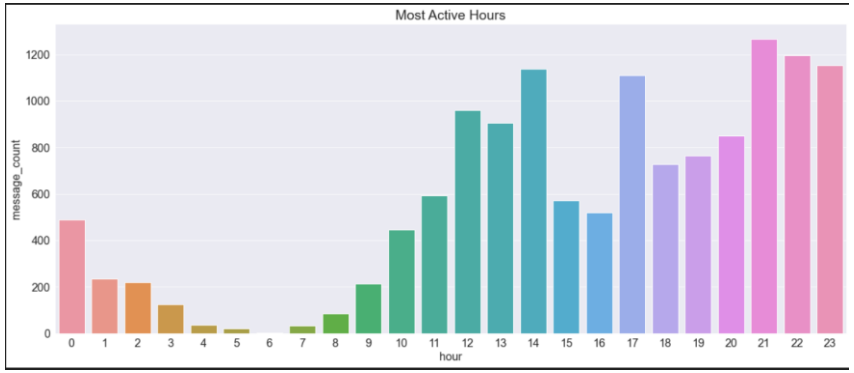
4. Activity Map

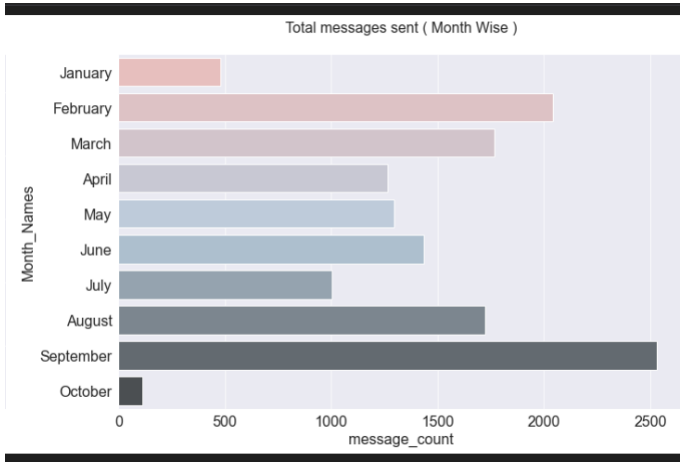
- Displaying engagement in form of Heat Map.
- Fetching day wise activity with respect to time as well.

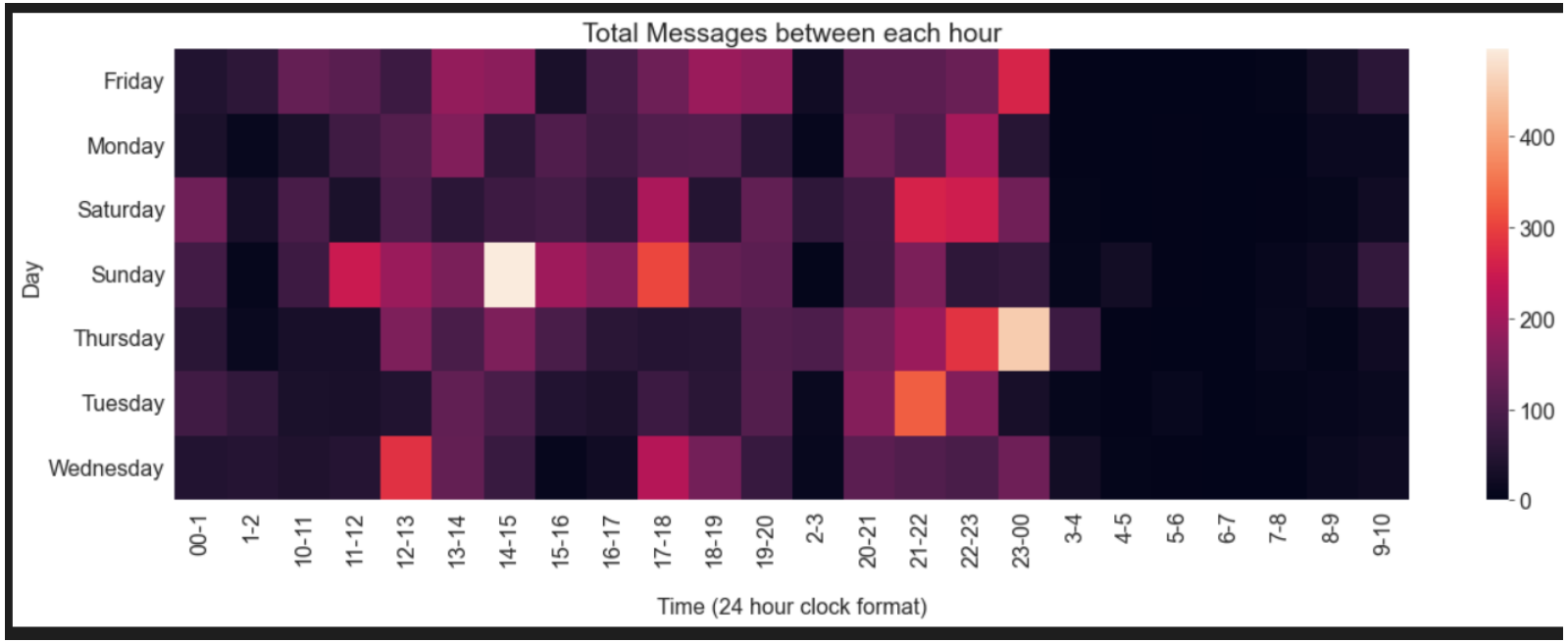
```
# fillna(0) means if no msg then replace NAN with 0
user_heatmap = df.pivot_table(index='day_name', columns='period', values='message', aggfunc='count').fillna(0)

plt.figure(figsize=(20,6))
#fig,ax = plt.subplots()
sns.heatmap(user_heatmap)
plt.yticks(rotation='horizontal')
plt.xlabel('\nTime (24 hour clock format)')
plt.ylabel('Day')
plt.title('Total Messages between each hour')
plt.show()
```

Output







EXPLORATORY DATA ANALYSIS-IV

5. Weekly and Monthly Activity

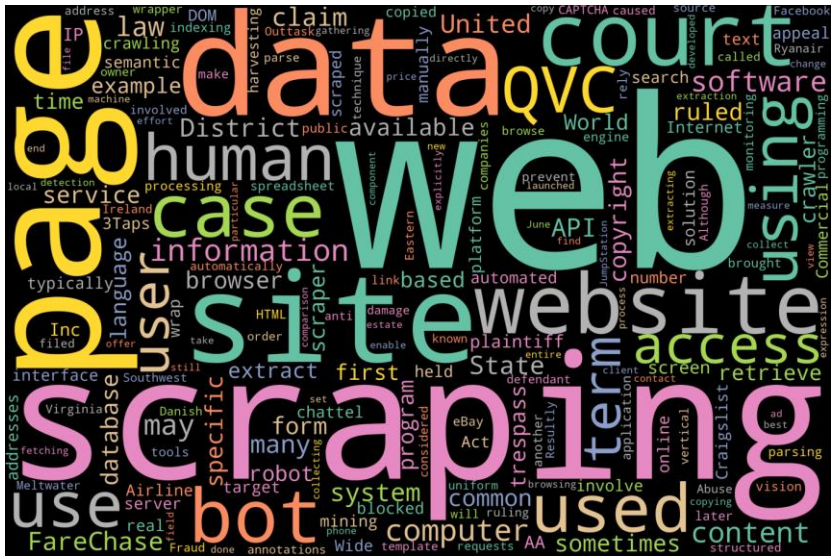
- Heatmap representation of monthly and weekly activity

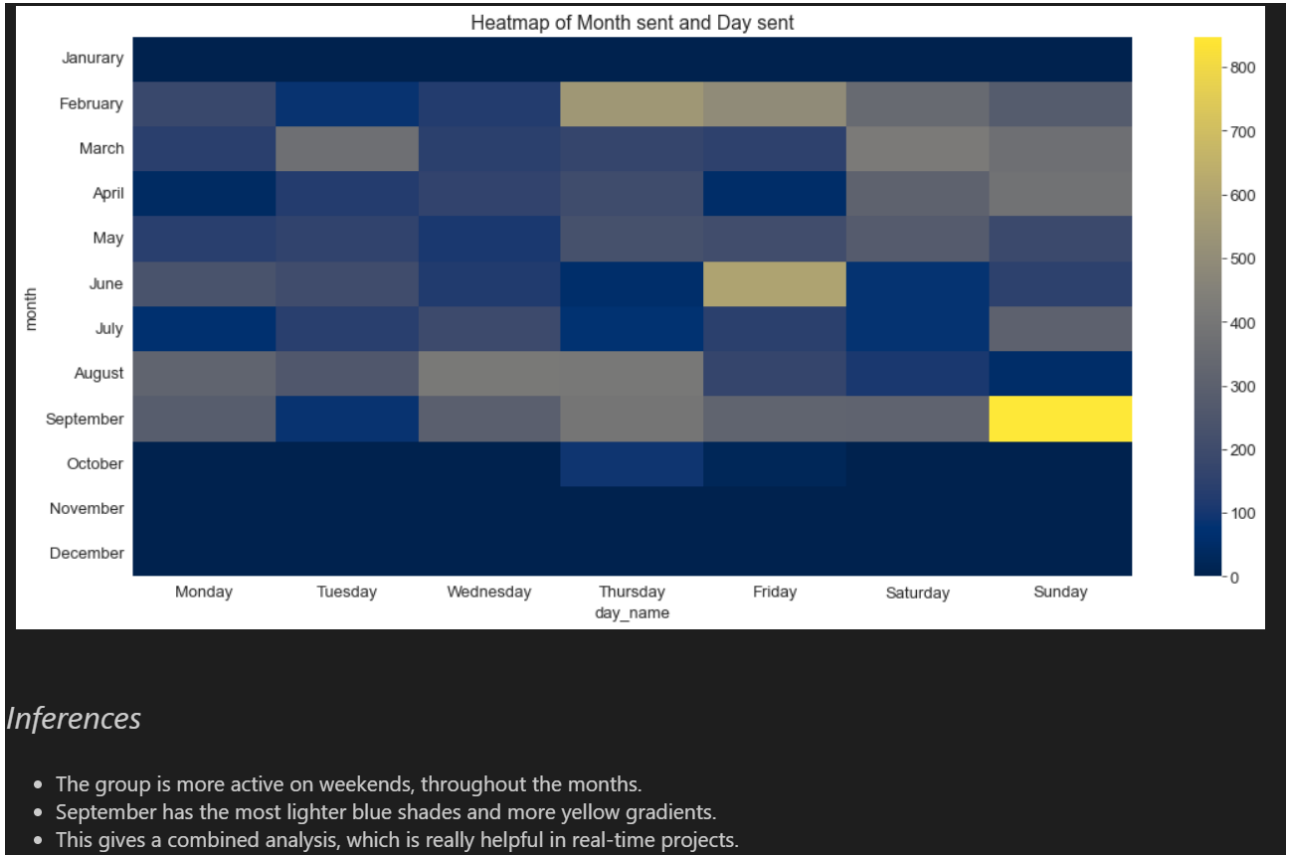
6. Stop Words & Word Cloud

```
# Making Word Cloud
temp = df[df['user'] != 'group_notification']
temp = temp[temp['message'] != '<Media omitted>\n']

wc = WordCloud(width=1500, height=1000,min_font_size=12,background_color='white')

# Image of Wordcloud
df_wc = wc.generate(temp['message'].str.cat(sep=" "))
fig,ax = plt.subplots()
ax.imshow(df_wc)
```





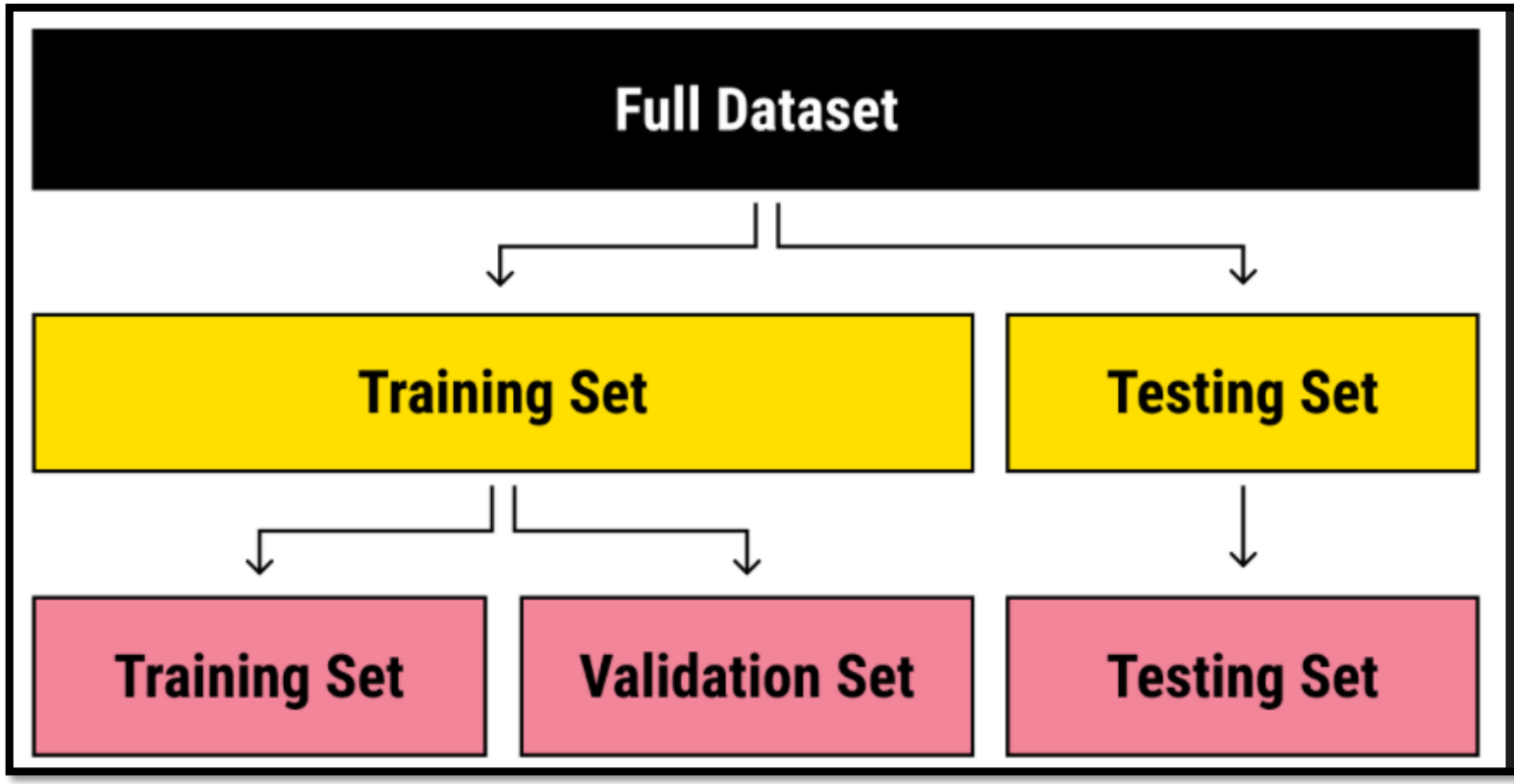
Inferences

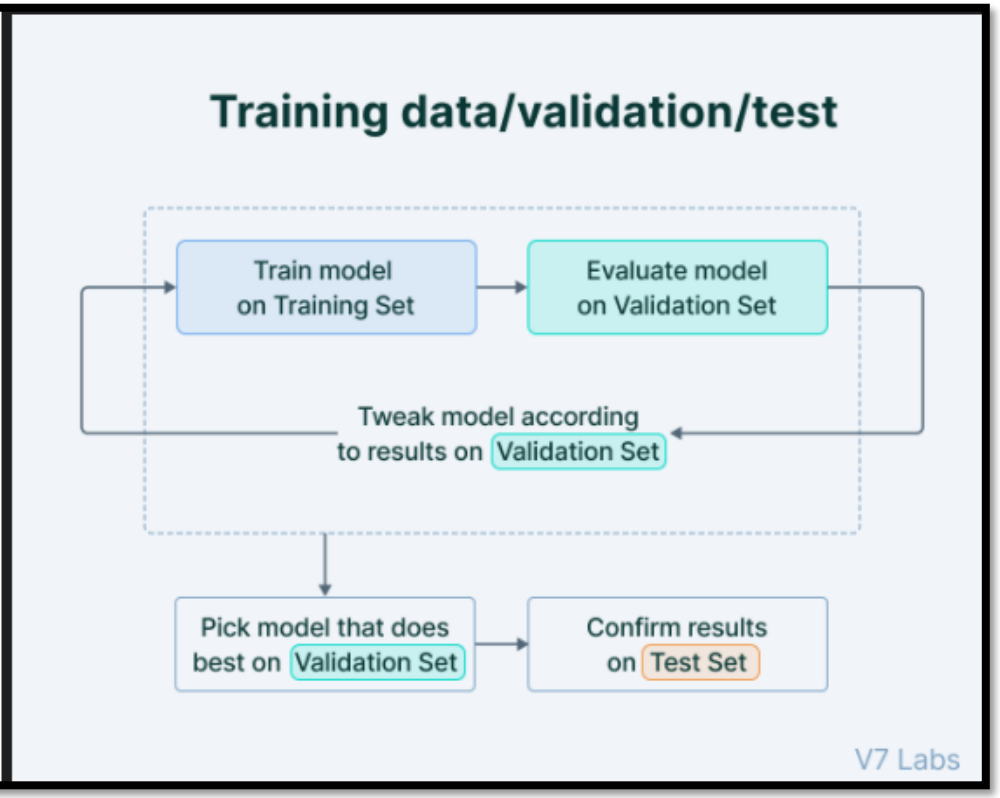
- The group is more active on weekends, throughout the months.
- September has the most lighter blue shades and more yellow gradients.
- This gives a combined analysis, which is really helpful in real-time projects.

LSTM MODULE

DATASET

- **Train.txt** - The model sees and learns from this data.
- **Val.txt** - Provides an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters.
- **Test.txt** - Provides an unbiased evaluation of a final model fit on the training dataset.





FLOW-CHART

Read Dataset

Pre-Processing

Label Encoding

Tokenization

Padding

Model Building

Activation Layer

Dense Layer

LSTM Layer

Max Pooling Layer

Convolution Layer

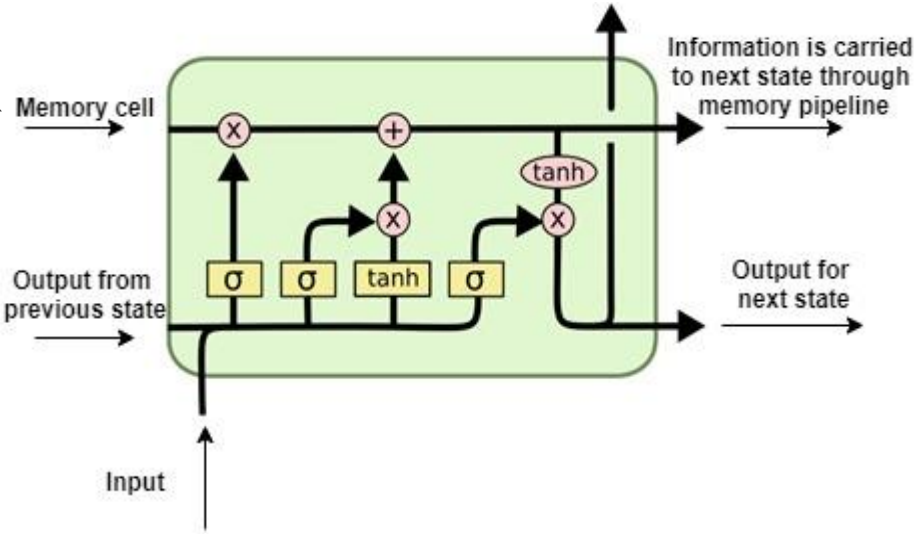
Drop Out Layer

Embedding Layer

Model Fitting

Evaluation

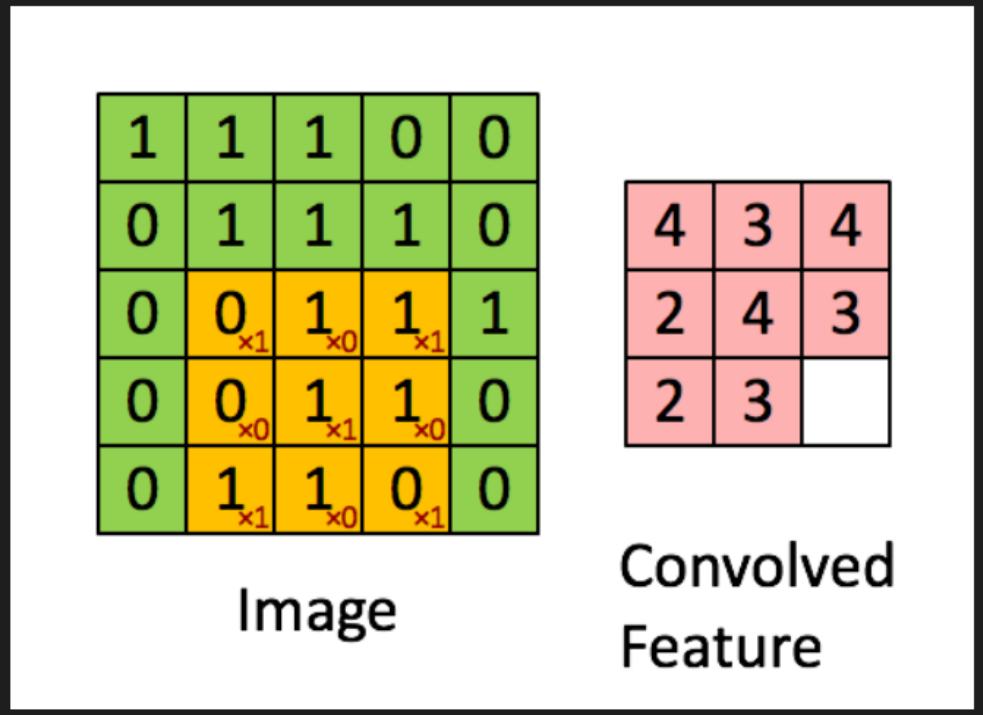
$$Ax = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \end{bmatrix}.$$



	the	cat	sat	on	mat
the	0	1	0	1	1
cat	1	0	1	0	0
sat	0	1	0	1	0
on	1	0	1	0	0
mat	1	0	0	0	0

LAYERS-I

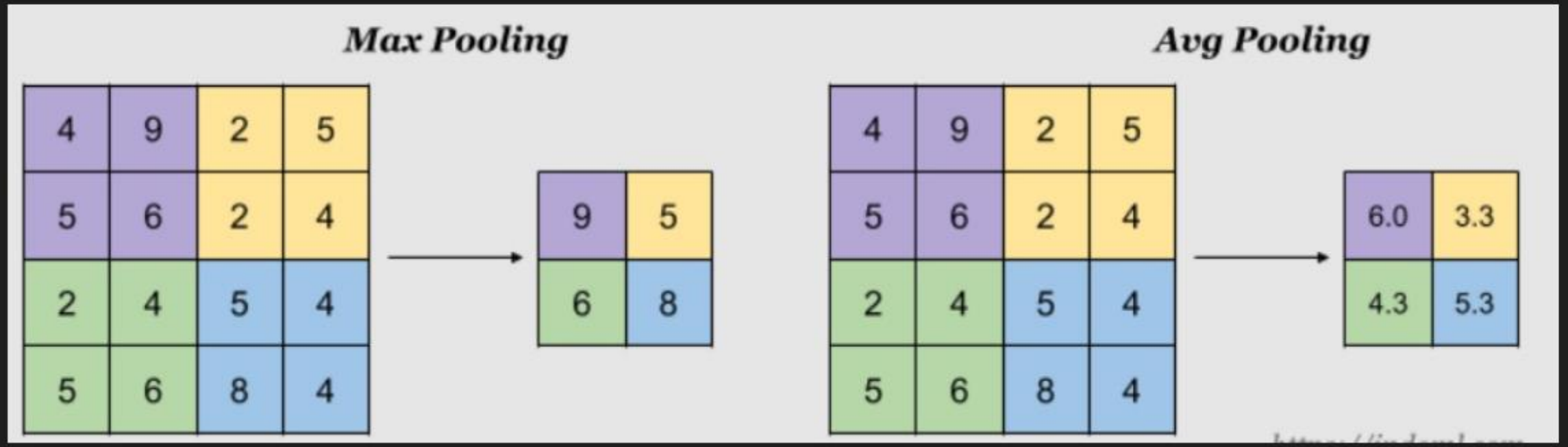
- The first layer is the **Embedded layer** that uses 200 length vectors to represent each word.
- The next layer is **Dropout layer**, i am using it because with the greater number of parameters there may be chance of causing overfitting. It prevents all neurons in a layer from synchronously optimizing their weights. It helps in increasing accuracy score. Here it can max dropout 25% bits.
- The next layer is **Convolution layer**. Mainly used for feature extraction with respect to filters.
  - In Case Kernel Size = 3 then Following will be feature extracted
  - A kernel is a small matrix, with its height and width smaller than the training dataset to be convolved. It is also known as a convolution matrix or convolution mask.



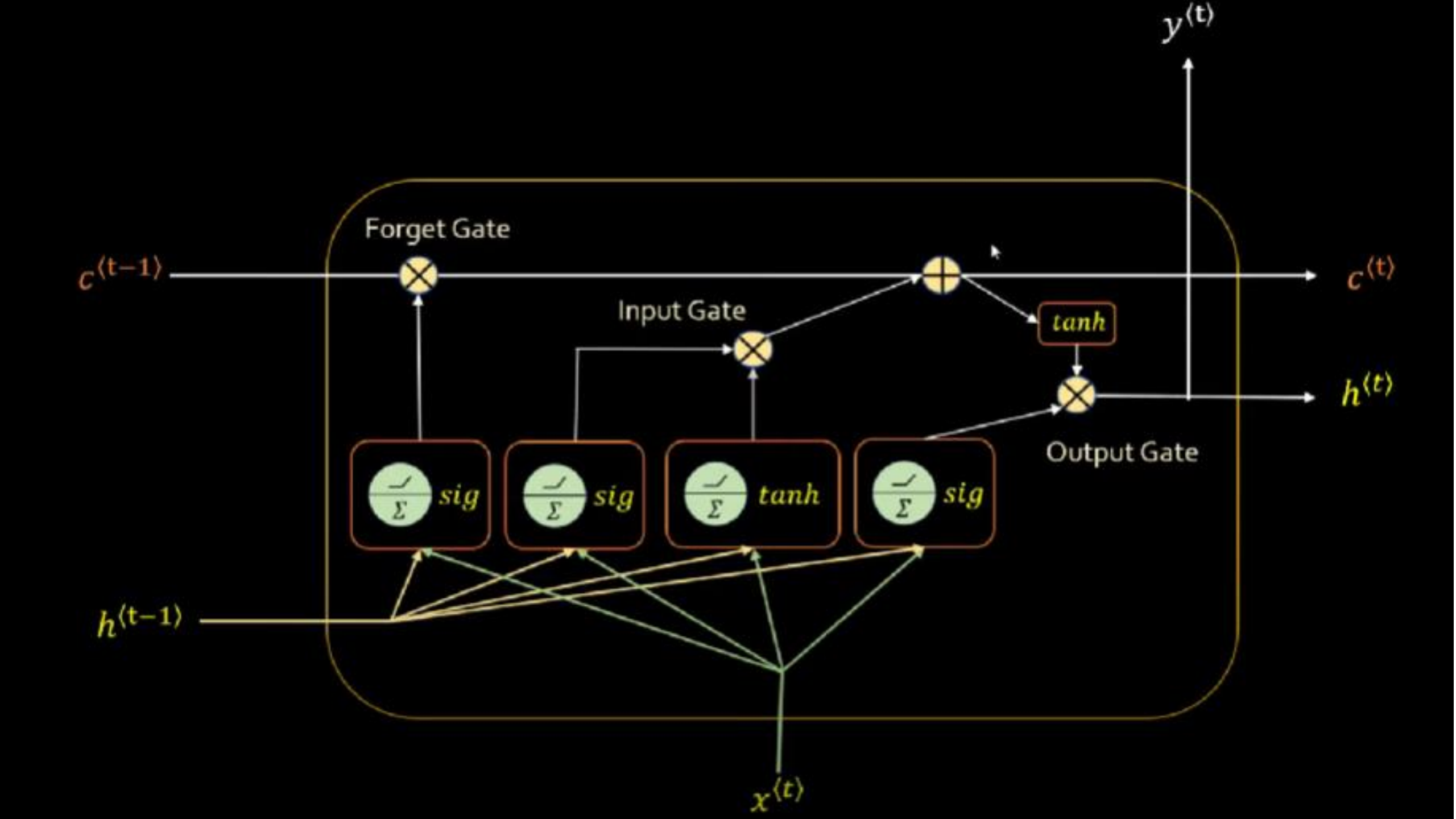
  -
  -
  - An activation function is the last component of the convolutional layer to increase the non-linearity in the output. Generally, ReLu function or Tanh function is used as an activation function in a convolution layer

LAYERS-II

- The next layer is **Maxpooling layer**. Pooling layer is used to reduce the size of the input image and speed up computation.



- The next layer is the **LSTM layer** with 128 memory units (smart neurons) that represent the dimensionality of outer space. It mainly used to classifying, processing and making predictions based on time series data
- The next layer is **Dense layer**. , Since this is a classification problem we use a Dense output layer with a single neuron and a sigmoid activation function to make 0 or 1 predictions for the each emotions.
- **Softmax** converts a vector of values to a probability distribution. The elements of the output vector are in range (0, 1) and sum to 1.


$y^{\langle t \rangle}$
Forget Gate
$c^{\langle t-1 \rangle}$
$c^{\langle t \rangle}$
Input Gate
tanh
$h^{\langle t \rangle}$
Output Gate
sig
sig
tanh
sig
$h^{\langle t-1 \rangle}$
$x^{\langle t \rangle}$

Formula for calculating Input Gate

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Where σ represents sigmoid function

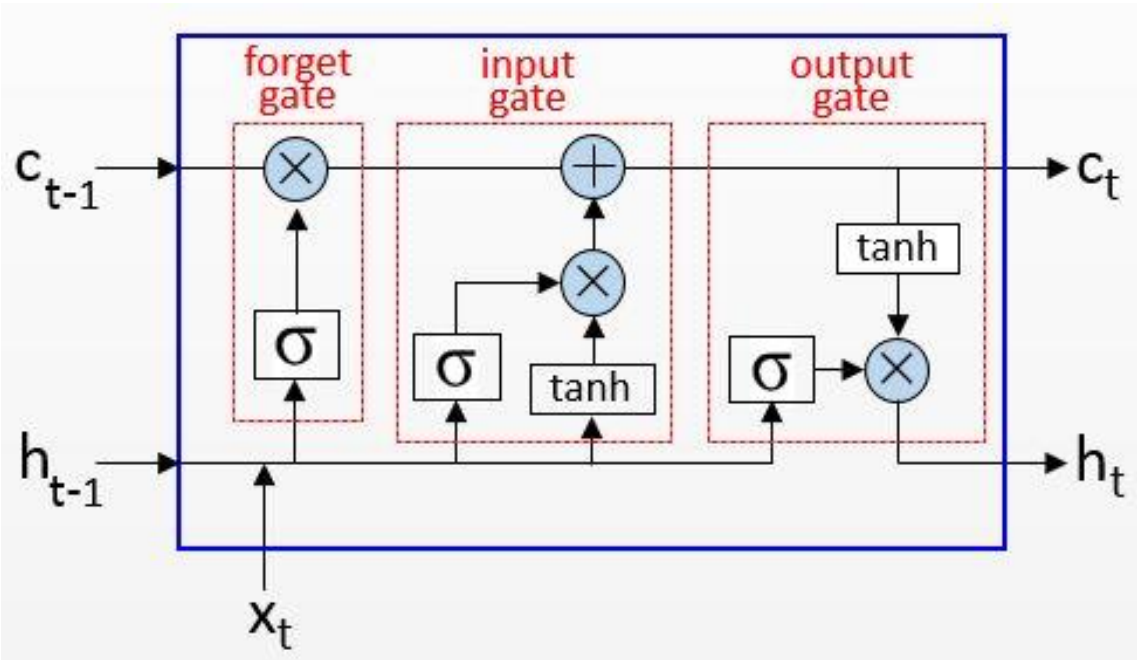
tanh represents tanh function

W_i is weight of current input gate neuron

h_{t-1} is output of the previous LSTM block (at timestamp t-1)

X_t is input at current timestamp (t)

b_i is biases for the input gates



Formula for calculating Forget Gate

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Where σ represents sigmoid function

F_t represents Forget Gate

W_f is weight of current Forget Gate neuron

h_{t-1} is output of the previous LSTM block (at timestamp t-1)

X_t is input at current timestamp (t)

B_f is biases for the Forget gate

Formula for calculating Output Gate

$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Where σ represents sigmoid function

O_t represents Output Gate

W_o is weight of current Output Gate neuron

h_{t-1} is output of the previous LSTM block (at timestamp t-1)

X_t is input at current timestamp (t)

B_o is biases for the Output gate

C_t Final calculated value at timestamp(t)

PRE-PROCESSSING

Pre-Processing of Dataset

```
# Text preprocessing function

str_punc = string.punctuation.replace(',', '').replace("'",'')

def clean(text):
    global str_punc
    text = re.sub(r'[^a-zA-Z ]', '', text)
    text = text.lower()
    return text
```

✓ 0.3s

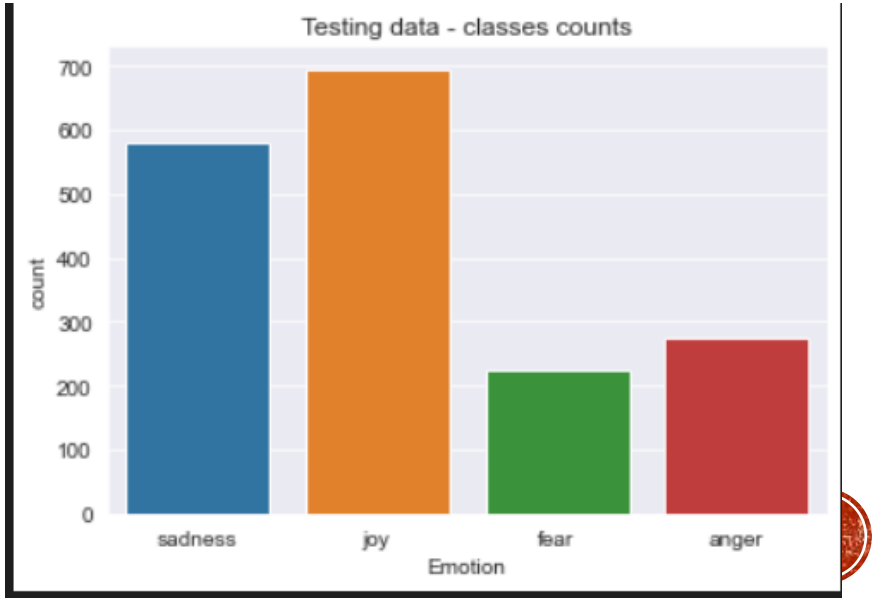
```
df_test = df_test[df_test['Emotion'].isin(['sadness','anger','joy','fear'])]
df_val = df_val[df_val['Emotion'].isin(['sadness','anger','joy','fear'])]
df_train = df_train[df_train['Emotion'].isin(['sadness','anger','joy','fear'])]

X_train = df_train['Text'].apply(clean)
y_train = df_train['Emotion']

X_test = df_test['Text'].apply(clean)
y_test = df_test['Emotion']

X_val = df_val['Text'].apply(clean)
y_val = df_val['Emotion']
```

✓ 0.3s



LABEL ENCODING

Label Encoding

- Label Encoding is a popular encoding technique for handling categorical variables.
- In this technique, each label is assigned a unique integer based on alphabetical ordering.

ID	Country	Population
1	Japan	127185332
2	U.S	326766748
3	India	1354051854
4	China	1415045928
5	U.S	326766748
6	India	1354051854

ID	Country	Population
1	0	127185332
2	1	326766748
3	2	1354051854
4	3	1415045928
5	1	326766748
6	2	1354051854

```
le = LabelEncoder()
y_train = le.fit_transform(y_train)
y_test = le.transform(y_test)
y_val = le.transform(y_val)
```

✓ 0.1s

```
y_train
```

✓ 0.1s

```
array([3, 3, 0, ..., 2, 0, 3])
```

- We use to_categorical to transform our training data
- If training data uses classes as numbers, to_categoric

```
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
y_val = to_categorical(y_val)
```

✓ 0.1s

```
y_train
```

✓ 0.2s

```
array([[0., 0., 0., 1.],
       [0., 0., 0., 1.],
       [1., 0., 0., 0.],
       ...,
       [0., 0., 1., 0.],
       [1., 0., 0., 0.],
       [0., 0., 0., 1.]], dtype=float32)
```

TEXT 2 SEQUENCE

we can use **'texts_to_sequences()'** method to assign integers to words in a list of sentence.

```
X_train.head()
```
[57] ✓ 0.1s

```
0                              i didnt feel humiliated
1    i can go from feeling so hopeless to so damned...
2     im grabbing a minute to post i feel greedy wrong
4                                i am feeling grouchy
5    ive been feeling a little burdened lately wasn...
Name: Text, dtype: object
```

```
sequences_train = tokenizer.texts_to_sequences(X_train)
sequences_test = tokenizer.texts_to_sequences(X_test)
sequences_val = tokenizer.texts_to_sequences(X_val)
```
[58] ✓ 0.8s

```
for i in range(3):
    print(X_train[i] ,'\n',sequences_train[i])
```
[59] ✓ 0.2s

```
i didnt feel humiliated
 [1, 137, 2, 581]
i can go from feeling so hopeless to so damned hopeful just from being around someone who cares and is awake
 [1, 39, 98, 58, 7, 14, 473, 4, 14, 4089, 486, 30, 58, 60, 127, 155, 75, 1576, 3, 21, 1210]
im grabbing a minute to post i feel greedy wrong
 [15, 2984, 6, 1149, 4, 292, 1, 2, 415, 383]
```

PADDING

```
X_train = pad_sequences(sequences_train, maxlen=256, truncating='pre') # truncating='pre' => remove post padding extra than maxlen
X_test = pad_sequences(sequences_test, maxlen=256, truncating='pre')
X_val = pad_sequences(sequences_val, maxlen=256, truncating='pre')

vocabSize = len(tokenizer.index_word) + 1
print(f"Vocabulary size = {vocabSize}")
```

0.3s

```
Vocabulary size = 14980
```

```
for i in range(2):
    print(X_train[i])
```

0.1s

```
[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   1 137   2 581]
```

MODEL BUILDING

```
print('Build model...')

model = Sequential()
model.add(Embedding(vocabSize, embedding_size, input_length=maxlen))
model.add(Dropout(0.25))
model.add(Conv1D(filters,
                 kernel_size,
                 padding='valid',
                 activation='relu',
                 strides=1))
model.add(MaxPooling1D(pool_size=pool_size))
model.add(LSTM(lstm_output_size))
model.add(Dense(4))
model.add(Activation('softmax'))

model.summary()
```

✓ 2.6s

MODEL SUMMARY

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 256, 200)          2996000

 dropout (Dropout)           (None, 256, 200)          0

 conv1d (Conv1D)             (None, 252, 128)          128128

 max_pooling1d (MaxPooling1D  (None, 63, 128)          0
 )

 lstm (LSTM)                 (None, 128)               131584

 dense (Dense)               (None, 4)                 516

 activation (Activation)     (None, 4)                 0

=================================================================
Total params: 3,256,228
Trainable params: 3,256,228
Non-trainable params: 0
```

MODEL FITTING

```
adam = Adam(learning_rate=0.005)
#model.compile(loss='mean_squared_error', optimizer=adam, metrics=['accuracy'])
#model.compile(loss='binary_crossentropy', optimizer=adam, metrics=['accuracy'])
model.compile(loss='categorical_crossentropy', optimizer=adam, metrics=['accuracy'])
```

✓ 0.2s

```
callback = EarlyStopping(
    monitor="val_loss",
    patience=3,
    restore_best_weights=True,
)
```

✓ 0.9s

Fitting the Model with Epochs = 40 , Batch size of 256

```
# Fit model
history = model.fit(X_train,
                    y_train,
                    validation_data=(X_val, y_val),
                    verbose=1,
                    batch_size=256,
                    epochs=40,
                    callbacks=[callback]
                   )
```

[67] ✓ 7m 54.7s

```
Epoch 1/40
56/56 [==============================] - 79s 1s/step - loss: 0.8550 - accuracy: 0.6296 - val_loss: 0.3940 - val_accuracy: 0.8639
Epoch 2/40
56/56 [==============================] - 76s 1s/step - loss: 0.1938 - accuracy: 0.9365 - val_loss: 0.1714 - val_accuracy: 0.9374
Epoch 3/40
56/56 [==============================] - 90s 2s/step - loss: 0.0617 - accuracy: 0.9790 - val_loss: 0.1664 - val_accuracy: 0.9426
Epoch 4/40
56/56 [==============================] - 77s 1s/step - loss: 0.0342 - accuracy: 0.9878 - val_loss: 0.1976 - val_accuracy: 0.9397
Epoch 5/40
56/56 [==============================] - 73s 1s/step - loss: 0.0229 - accuracy: 0.9918 - val_loss: 0.1741 - val_accuracy: 0.9408
Epoch 6/40
56/56 [==============================] - 79s 1s/step - loss: 0.0146 - accuracy: 0.9947 - val_loss: 0.1887 - val_accuracy: 0.9437
```

EVALUATION

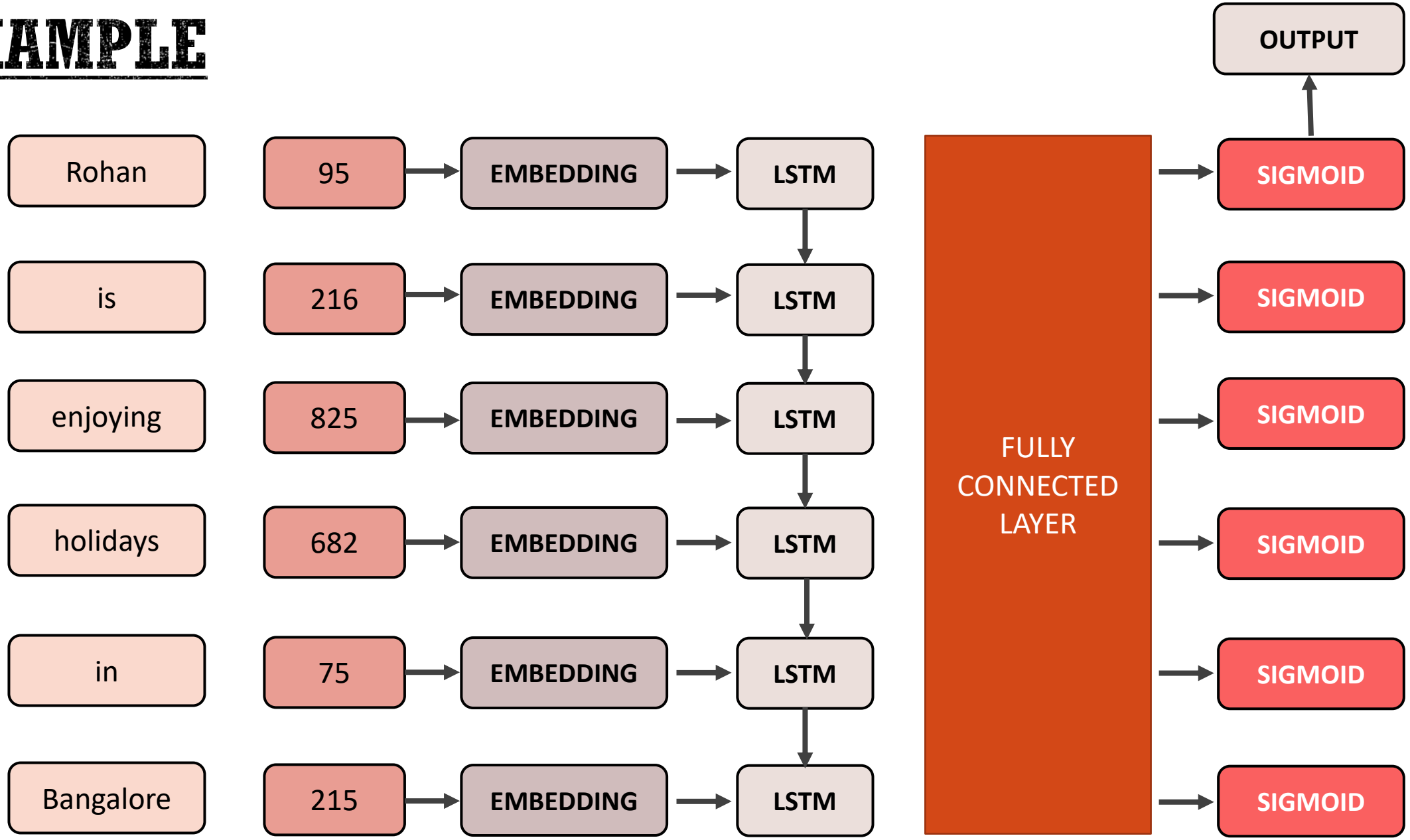
- verbose=0 will show nothing (silent)
- verbose=1 will show an animated progress bar like this:
- [==============================]
- verbose=2 will just mention the number of epoch like this: Epoch 1/50

```
scores = model.evaluate(X_test, y_test, verbose=1)
print("Accuracy : %.2f%%" % (scores[1]*100))
```

✓ 2.8s

```
56/56 [==============================] - 3s 47ms/step - loss: 0.1696 - accuracy: 0.9454
Accuracy : 94.54%
```

EXAMPLE


OUTPUT
Rohan
95
EMBEDDING
LSTM
SIGMOID
is
216
EMBEDDING
LSTM
SIGMOID
enjoying
825
EMBEDDING
LSTM
SIGMOID
holidays
682
EMBEDDING
LSTM
SIGMOID
in
75
EMBEDDING
LSTM
SIGMOID
Bangalore
215
EMBEDDING
LSTM
SIGMOID
FULLY CONNECTED LAYER

THANK YOU