

# **DEEP LEARNING ASSIGNMENT**

## **Mini Project on Natural Language Processing**

Akash.E  
2018103005  
CSE - 'P' Batch  
21/05/2021

# **CS-6005 DEEP LEARNING ASSIGNMENT - 5**

AKASH.E - (2018103005)

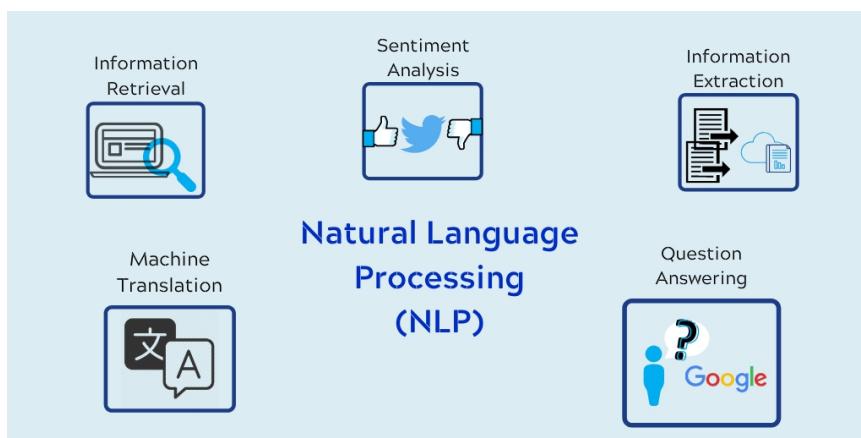
## **NATURAL LANGUAGE PROCESSING**

### **1. INTRODUCTION:**

In this project we try to do sentiment analysis on Covid-19 tweets using Natural Language Processing tools and Bi-Directional LSTM.

Natural language processing (NLP) refers to the branch of computer science and more specifically, the branch of artificial intelligence or AI concerned with giving computers the ability to understand text and spoken words in much the same way human beings can.

NLP combines computational linguistics rule-based modeling of human language with statistical, machine learning, and deep learning models. Together, these technologies enable computers to process human language in the form of text or voice data and to understand its full meaning, complete with the speaker or writer's intent and sentiment.

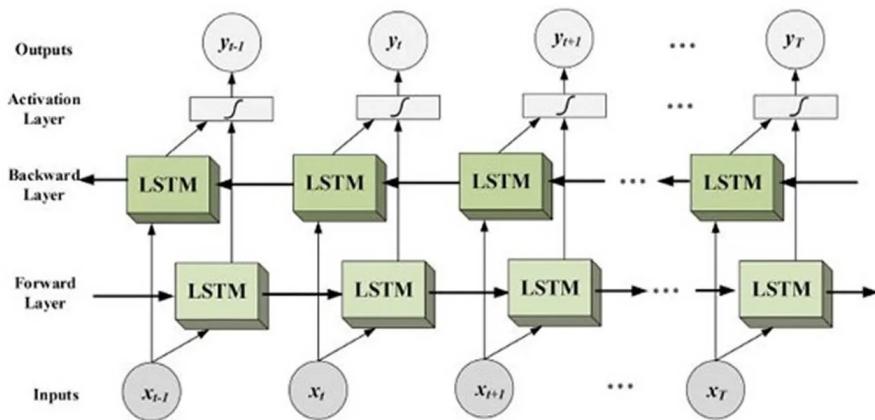


The Python programming language provides a wide range of tools and libraries for attacking specific NLP tasks. Many of these are found in the Natural Language Toolkit, or NLTK, an open source collection of libraries, programs, and education resources for building NLP programs.

## **2. BI-DIRECTIONAL LSTM :**

Bidirectional LSTMs are an extension of traditional LSTMs that can improve model performance on sequence classification problems.

In problems where all timesteps of the input sequence are available, Bidirectional LSTMs train two instead of one LSTMs on the input sequence. This can provide additional context to the network and result in faster and even fuller learning on the problem.



The idea of Bidirectional Recurrent Neural Networks (RNNs) involves duplicating the first recurrent layer in the network so that there are now two layers side-by-side, then providing the input sequence as-is as input to the first layer and providing a reversed copy of the input sequence to the second.

This approach has been used to great effect with Long Short-Term Memory (LSTM) Recurrent Neural Networks. The use of providing the sequence bi-directionally was initially justified in the domain of speech recognition because there is evidence that the context of the whole utterance is used to interpret what is being said rather than a linear interpretation.

In Bi-LSTM you preserve information from the future and using the two hidden states combined you are able in any point of time to preserve information from both **past and future**.

### **3. DATASET :**

The dataset consists of tweets related to Covid-19. Data is extracted from twitter accounts of relevant most followed accounts using Twitter API. The accounts include, but are not limited to news sources, medical organizations, politicians and relevant ministers. Dataset consists of the following columns :

- 1) Location
- 2) Tweet At
- 3) Original Tweet
- 4) Label

In the upcoming modules we will perform some EDA on the dataset for pre-processing. Below is a small glimpse of the dataset.

# UserName	# ScreenNa...	▲ Location	▲ TweetAt	▲ OriginalTw...	▲ Sentiment
1	44953	NYC	02-03-2020	TRENDING: New Yorkers encounter empty supermarket shelves (pictured, Wegmans in Brooklyn), sold-out ...	Extremely Negative
2	44954	Seattle, WA	02-03-2020	When I couldn't find hand sanitizer at Fred Meyer, I turned to #Amazon. But \$114.97 for a 2 pack of ...	Positive
3	44955		02-03-2020	Find out how you can protect yourself and loved ones from #coronavirus. ?	Extremely Positive

The target value is the ‘Sentiment’ column which has 5 classes namely Positive, Negative, Neutral, Extremely Positive, Extremely Negative.

## **4. MODULE LIST :**

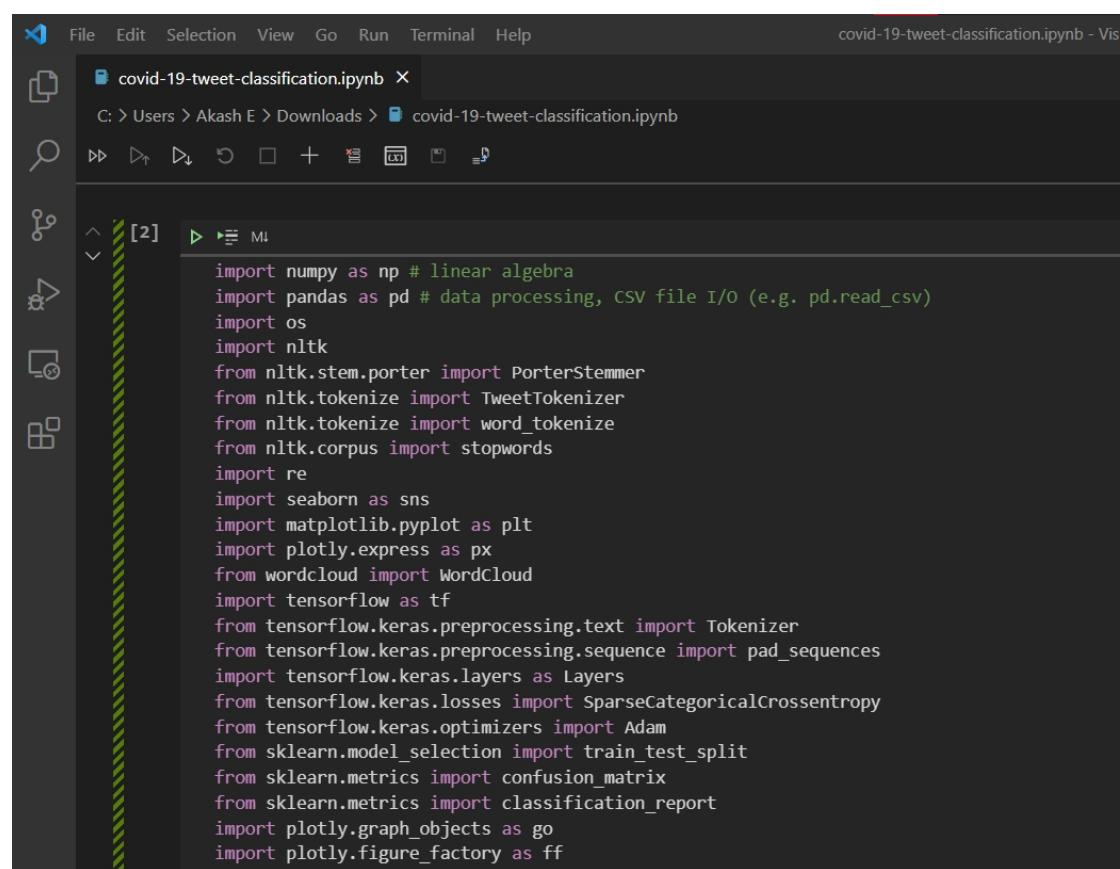
1. Exploratory Data Analysis.
2. Data Pre-Processing.
3. Word Embedding.
4. Model Building and Training.
5. Result Analysis.

## **5. MODULE ELABORATION:**

### **5.1 Exploratory Data Analysis :**

In this module we import the data to our project and do some data visualization and exploration.

First, we import all the necessary libraries :



The screenshot shows a Jupyter Notebook interface with a dark theme. The top bar includes File, Edit, Selection, View, Go, Run, Terminal, Help, and a tab for 'covid-19-tweet-classification.ipynb - Vis'. The left sidebar has icons for file operations like Open, Save, and Run Cell. The main area shows a code cell labeled [2] containing Python imports for various libraries including numpy, pandas, os, nltk, PorterStemmer, TweetTokenizer, word\_tokenize, stopwords, re, seaborn, plt, px, WordCloud, tensorflow, Tokenizer, pad\_sequences, Layers, SparseCategoricalCrossentropy, Adam, train\_test\_split, confusion\_matrix, classification\_report, go, and ff.

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os
import nltk
from nltk.stem.porter import PorterStemmer
from nltk.tokenize import TweetTokenizer
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
import re
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
from wordcloud import WordCloud
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
import tensorflow.keras.layers as Layers
from tensorflow.keras.losses import SparseCategoricalCrossentropy
from tensorflow.keras.optimizers import Adam
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
import plotly.graph_objects as go
import plotly.figure_factory as ff
```

## Importing the dataset :

```
  covid-19-tweet-classification.ipynb ×
C: > Users > Akash E > Downloads > covid-19-tweet-classification.ipynb
▶ [3] ▶ ML
train=pd.read_csv("C:/Users/Aakash_E/Downloads/input/covid-19-nlp-text-classification/Corona_NLP_train.csv",encoding='latin1')

[4] ▶ ML
train.head()

  UserName ScreenName Location TweetAt OriginalTweet Sentiment
0 3799 48751 London 16-03-2020 @MeNyrbie @Phil_Gahan @Chrisitv https://t.co/i...
1 3800 48752 UK 16-03-2020 advice Talk to your neighbours family to excha...
2 3801 48753 Vagabonds 16-03-2020 Coronavirus Australia: Woolworths to give elde...
3 3802 48754 NaN 16-03-2020 My food stock is not the only one which is emp...
4 3803 48755 NaN 16-03-2020 Me, ready to go at supermarket during the #COV... Extremely Negative

[5] ▶ ML
train['Location'].unique()

array(['London', 'UK', 'Vagabonds', ..., 'Juba south sudan', 'OHIO',
       'i love you so much || he/him'], dtype=object)
```

```
  covid-19-tweet-classification.ipynb ×
C: > Users > Akash E > Downloads > covid-19-tweet-classification.ipynb
▶ [6] ▶ ML
train['Location'].value_counts()

London 540
United States 528
London, England 520
New York, NY 395
Washington, DC 373
...
East Ferris 1
Wirral UK 1
?????? 1
USA, Europe, Brazil 1
California ? Washington DC 1
Name: Location, Length: 12220, dtype: int64

[7] ▶ ML
train.isnull().sum()

UserName 0
ScreenName 0
Location 8590
TweetAt 0
OriginalTweet 0
Sentiment 0
dtype: int64

[8] ▶ ML
train.shape

(41157, 6)
```

```

covid-19-tweet-classification.ipynb ×
C: > Users > Akash E > Downloads > covid-19-tweet-classification.ipynb

[9] train.describe()

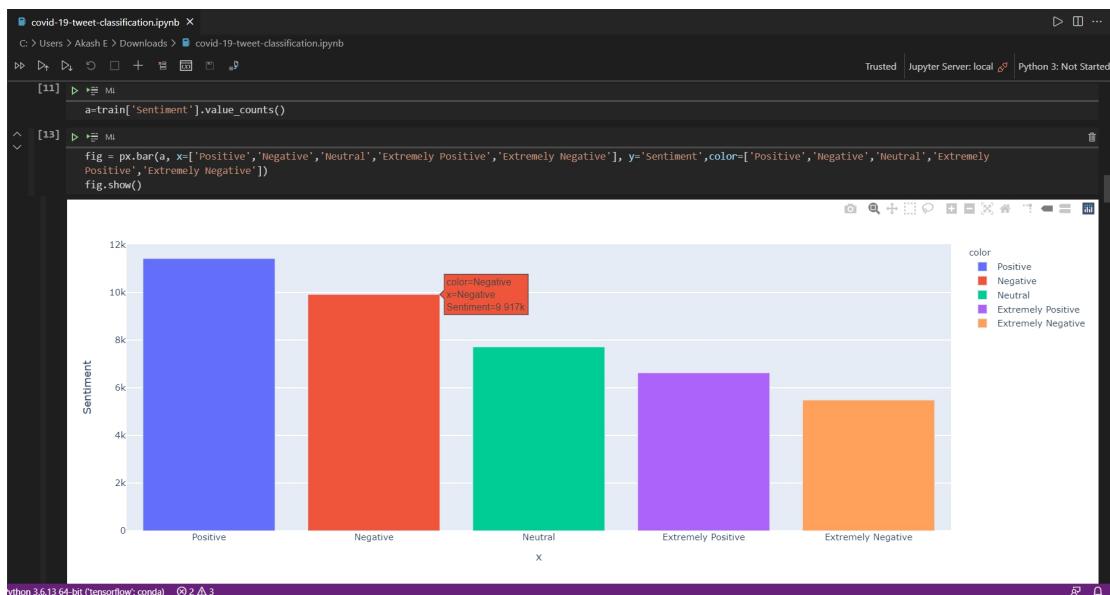
    UserName      ScreenName
count  41157.000000  41157.000000
mean   24377.000000  69329.000000
std    11881.146851  11881.146851
min    3799.000000  48751.000000
25%   14088.000000  59040.000000
50%   24377.000000  69329.000000
75%   34666.000000  79618.000000
max   44955.000000  89907.000000

[10] train.info()

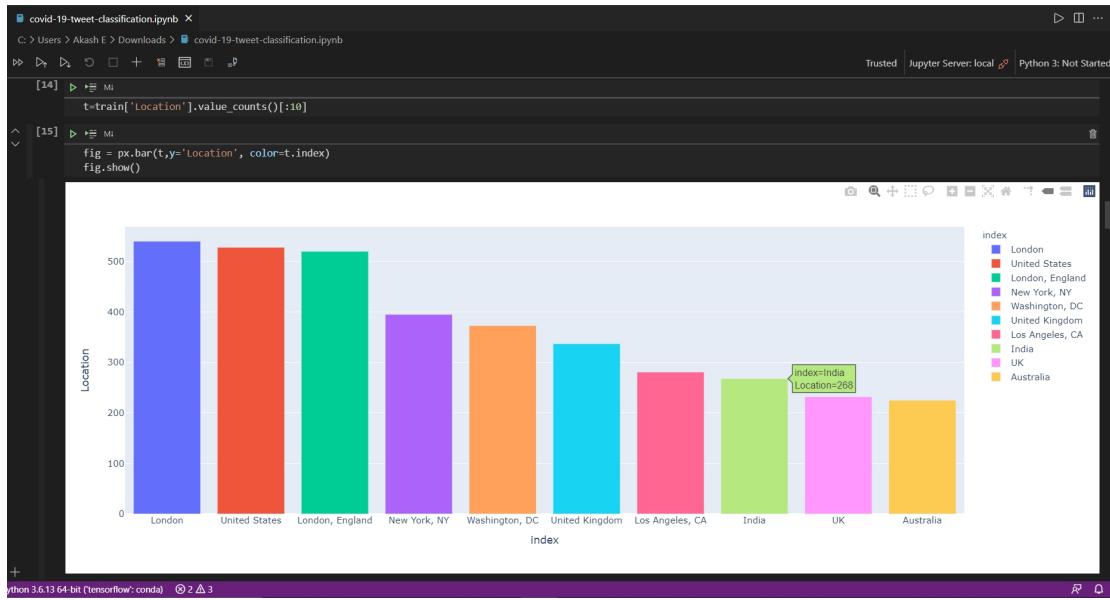
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41157 entries, 0 to 41156
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   UserName    41157 non-null   int64  
 1   ScreenName  41157 non-null   int64  
 2   Location    32567 non-null   object  
 3   TweetAt     41157 non-null   object  
 4   OriginalTweet  41157 non-null   object  
 5   Sentiment   41157 non-null   object  
dtypes: int64(2), object(4)
memory usage: 1.9+ MB

```

## Count of tweets for each class :



## Count of tweets in each location :



```
[14] > px.MI
t=train['Location'].value_counts()[:10]
[15] > px.MI
fig = px.bar(t,y='location', color=t.index)
fig.show()
```

```
[17] > px.MI
time=train['TweetAt'].to_frame()
[18] > px.MI
y=train['Sentiment'].copy()
[19] > px.MI
train['Sentiment']=train['Sentiment'].map({'Positive':0,'Negative':1,'Neutral':2,'Extremely Positive':3,
                                         'Extremely Negative':4})
[20] > px.MI
train['OriginalTweet']

0      @Merrybie @Phil Gahan @Chrisity https://t.co/i...
1      advice Talk to your neighbours family to excha...
2      Coronavirus Australia: Woolworths to give elde...
3      My food stock is not the only one which is emp...
4      Me, ready to go at supermarket during the #COV...
41152   Airline pilots offering to stock supermarket s...
41153   Response to complaint not provided citing COV...
41154   You know it's getting tough when @CameronWild...
41155   Is it wrong that the same of hand sanitizer i...
41156   @Artificat Well @Cameron Wild s are going for ...
Name: OriginalTweet, Length: 41157, dtype: object
```

```
[21] > px.MI
train['OriginalTweet'][0]
```

```
'@Merrybie @Phil Gahan @Chrisity https://t.co/ifz9FAn2Pa and https://t.co/xX6ghGFzCC and https://t.co/I2N1zdxNo8'
```

## 5.2 Data Pre-Processing :

In this module we clean the data to get only the necessary information.

## Function to remove urls,hastags,digits,html tags etc :

```
  covid-19-tweet-classification.ipynb X
C: > Users > Akash E > Downloads > covid-19-tweet-classification.ipynb
[23] ▶ ➜ ML
def cleaner(tweet):

    # remove urls
    tweet = re.sub(r'http\S+', ' ', tweet)

    # remove html tags
    tweet = re.sub(r'<.*?>', ' ', tweet)

    # remove digits
    tweet = re.sub(r'\d+', ' ', tweet)

    # remove hashtags
    tweet = re.sub(r'#\w+', ' ', tweet)

    # remove mentions
    tweet = re.sub(r'@\w+', ' ', tweet)

    #removing stop words
    tweet = tweet.split()
    tweet = " ".join([word for word in tweet if not word in stop_words])

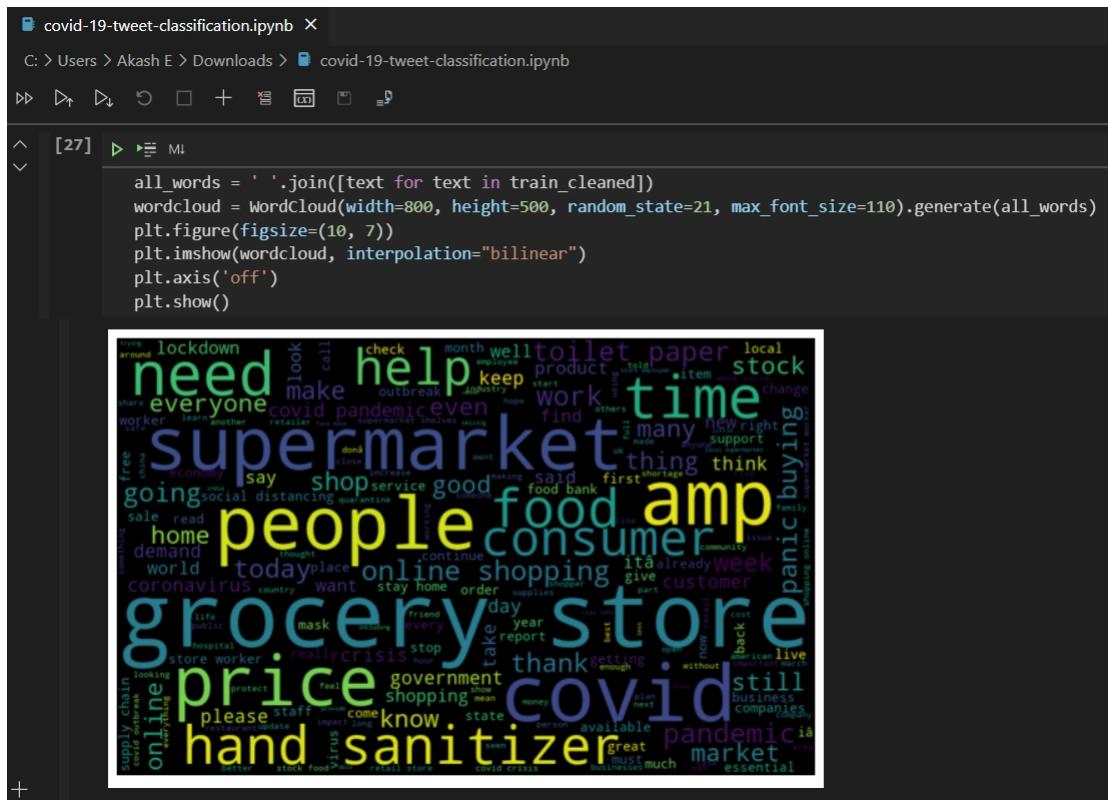
return tweet
```

```
  covid-19-tweet-classification.ipynb X
C: > Users > Akash E > Downloads > covid-19-tweet-classification.ipynb
[24] ▶ ➜ ML
stop_words = stopwords.words('english')
train['OriginalTweet']=train['OriginalTweet'].apply(lambda x:x.lower())
train['OriginalTweet'] = train['OriginalTweet'].apply(lambda x: ' '.join([w for w in x.split() if len(w)>3]))
train_cleaned = train['OriginalTweet'].apply(cleaner)
train_cleaned.head()

0    advice talk neighbours family exchange phone n...
1    coronavirus australia: woolworths give elderly...
2    food stock empty... please, panic, enough food...
3    ready supermarket outbreak. paranoid, food sto...
4    Name: OriginalTweet, dtype: object

[25] ▶ ➜ ML
train['OriginalTweet']=train_cleaned
```

Word cloud of all the words in cleaned data :

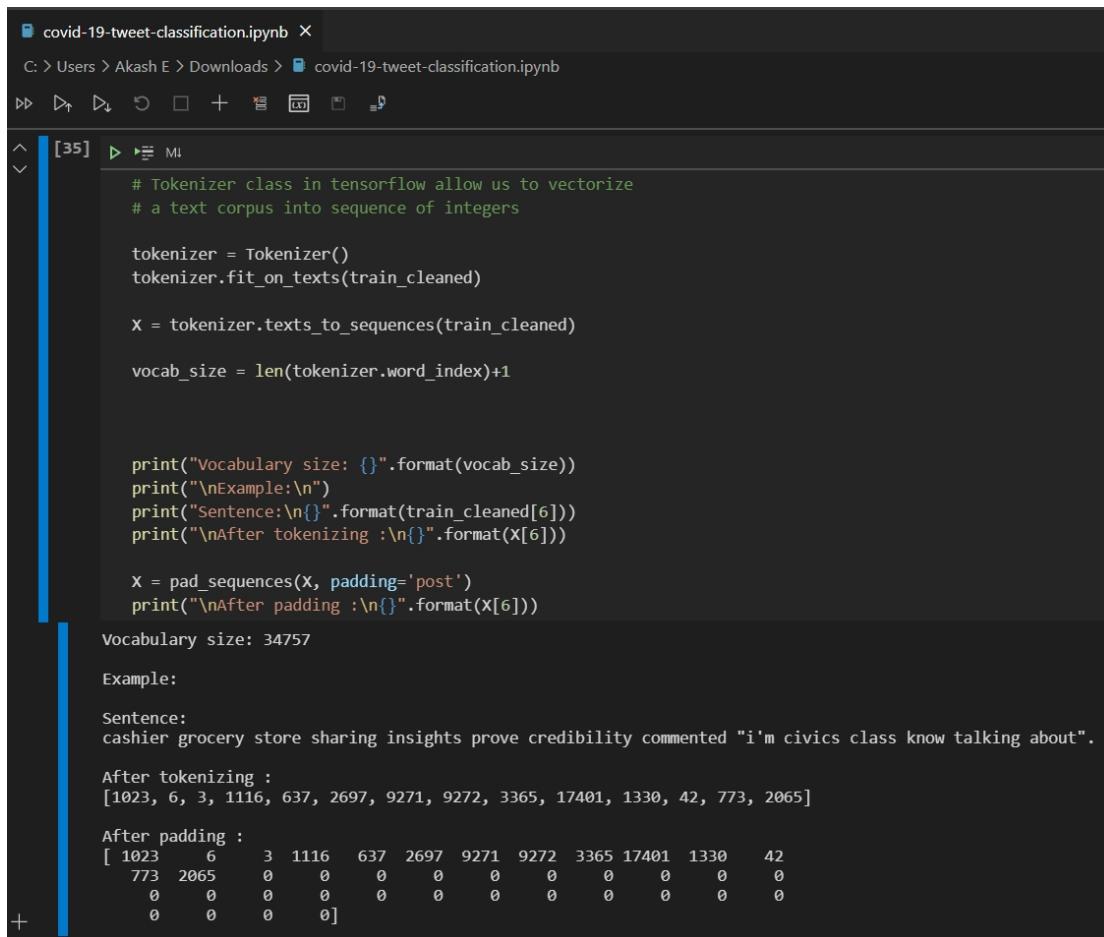


## 5.3 Word Embedding :

A word embedding is a learned representation for text where words that have the same meaning have a similar representation. Word embeddings are in fact a class of techniques where individual words are represented as real-valued vectors in a predefined vector space. Each word is mapped to one vector and the vector values are learned in a way that resembles a neural network, and hence the technique is often lumped into the field of deep learning.

Each of our word in the cleaned data is first mapped to a dictionary and the value of the index in the dictionary is given to the word. Then each sentence is padded with zeros in the end to keep every sentence the same size. This input is given to our **Keras Embedding layer** to perform the word embedding.

## Tokenizing and Padding each sentence :



```
[35]: # Tokenizer class in tensorflow allow us to vectorize
      # a text corpus into sequence of integers

tokenizer = Tokenizer()
tokenizer.fit_on_texts(train_cleaned)

X = tokenizer.texts_to_sequences(train_cleaned)

vocab_size = len(tokenizer.word_index)+1

print("Vocabulary size: {}".format(vocab_size))
print("\nExample:")
print("Sentence:\n{}".format(train_cleaned[6]))
print("\nAfter tokenizing :\n{}".format(X[6]))

X = pad_sequences(X, padding='post')
print("\nAfter padding :\n{}".format(X[6]))
```

Vocabulary size: 34757  
Example:  
Sentence:  
cashier grocery store sharing insights prove credibility commented "i'm civics class know talking about".  
After tokenizing :  
[1023, 6, 3, 1116, 637, 2697, 9271, 9272, 3365, 17401, 1330, 42, 773, 2065]  
After padding :  
[ 1023 6 3 1116 637 2697 9271 9272 3365 17401 1330 42  
 773 2065 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 ]

Labels are renamed as just Positive, Negative and Neutral :



```
[36]: encoding = {'Extremely Negative': 0,
               'Negative': 0,
               'Neutral': 1,
               'Positive':2,
               'Extremely Positive': 2
               }

labels = ['Negative', 'Neutral', 'Positive']

y.replace(encoding, inplace=True)
```

## 5.4 Model Building and Training :

We build our model with the following parameters and train it in this module.

Parameter	Value
Epochs	2
Batch Size	32
Embedding Dimension	16
Loss Function	Sparse Categorical Cross-Entropy
Learning Rate	0.01
Optimizer	Adam

```

# covid-19-tweet-classification.ipynb ×
C: > Users > Akash E > Downloads > covid-19-tweet-classification.ipynb

[37] tf.keras.backend.clear_session()

# hyper parameters
EPOCHS = 2
BATCH_SIZE = 32
embedding_dim = 16
units = 256

model = tf.keras.Sequential([
    Layers.Embedding(vocab_size, embedding_dim, input_length=x.shape[1]),
    Layers.Bidirectional(Layers.LSTM(units, return_sequences=True)),
    Layers.GlobalMaxPool1D(),
    Layers.Dropout(0.4),
    Layers.Dense(64, activation="relu"),
    Layers.Dropout(0.4),
    Layers.Dense(3)
])

model.compile(loss=SparseCategoricalCrossentropy(from_logits=True),
              optimizer='adam', metrics=['accuracy'])

model.summary()

```

## Model summary :

```

Model: "sequential"
-----  

Layer (type)          Output Shape       Param #
-----  

embedding (Embedding) (None, 40, 16)      556112  

bidirectional (Bidirectional) (None, 40, 512) 559104  

global_max_pooling1d (Global) (None, 512)     0  

dropout (Dropout)      (None, 512)        0  

dense (Dense)          (None, 64)         32832  

dropout_1 (Dropout)     (None, 64)         0  

dense_1 (Dense)        (None, 3)          195  

-----  

Total params: 1,148,243  

Trainable params: 1,148,243  

Non-trainable params: 0
-----
```

## Training the model :



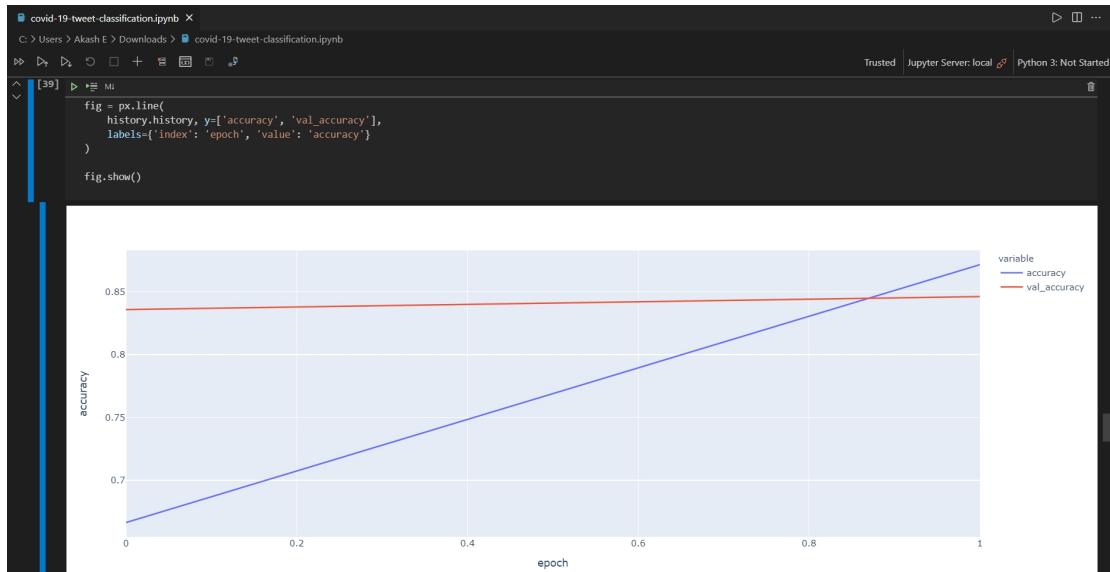
```
covid-19-tweet-classification.ipynb ×
C:\Users\Akash E\Downloads> covid-19-tweet-classification.ipynb
[38] ML
history = model.fit(X, y, epochs=EPOCHS, validation_split=0.12, batch_size=BATCH_SIZE)
Epoch 1/2
1132/1132 [=====] - 18s 15ms/step - loss: 0.7401 - accuracy: 0.6664 - val_loss: 0.4745 - val_accuracy: 0.8358
Epoch 2/2
1132/1132 [=====] - 17s 15ms/step - loss: 0.3845 - accuracy: 0.8715 - val_loss: 0.4231 - val_accuracy: 0.8461
```

After two epochs, our model's validation accuracy seem to decrease. Hence we stop the training at the second epoch. If trained for 10 epochs, our model has a good training accuracy but performs poorly with test data. We got a training accuracy of 87%

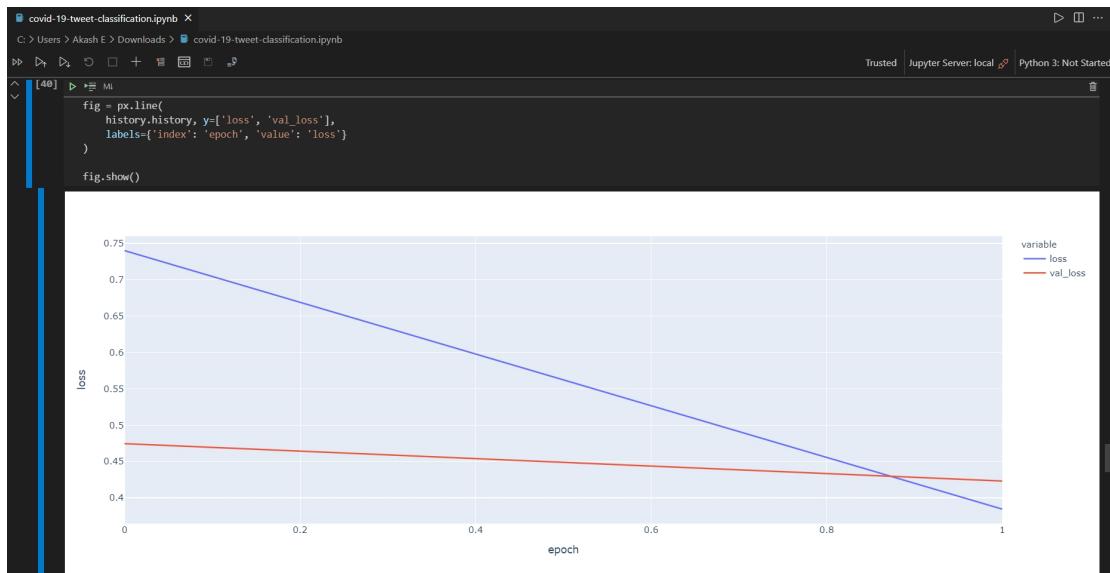
## 5.5 Result Analysis :

Here, we plot the training and testing results of our model.

### Accuracy plot :



## Loss plot :



## Importing the testing data :

```
[41]: test=pd.read_csv("../input/covid-19-nlp-text-classification/Corona_NLP_test.csv")  
[42]: test.head()  
  
   UserName ScreenName      Location TweetAt OriginalTweet Sentiment  
0         1     44953        NYC 02-03-2020 TRENDING: New Yorkers encounter empty supermar... Extremely Negative  
1         2     44954  Seattle, WA 02-03-2020 When I couldn't find hand sanitizer at Fred Me... Positive  
2         3     44955        NaN 02-03-2020 Find out how you can protect yourself and love... Extremely Positive  
3         4     44956  Chicagoland 02-03-2020 #Panic buying hits #NewYork City as anxious sh... Negative  
4         5     44957  Melbourne, Victoria 03-03-2020 #toiletpaper #dunnypaper #coronavirus #coronav... Neutral  
  
[43]: test.shape  
(3798, 6)
```

```

covid-19-tweet-classification.ipynb ×
C: > Users > Akash E > Downloads > covid-19-tweet-classification.ipynb

[44] ▶ ML
x_test = test['OriginalTweet'].copy()
y_test = test['sentiment'].copy()

x_test = x_test.apply(cleaner)

x_test = tokenizer.texts_to_sequences(x_test)

x_test = pad_sequences(x_test, padding='post')

y_test.replace(encoding, inplace=True)

[45] ▶ ML
pred = model.predict_classes(x_test)

[46] ▶ ML
pred

array([0, 2, 2, ..., 1, 0, 2])

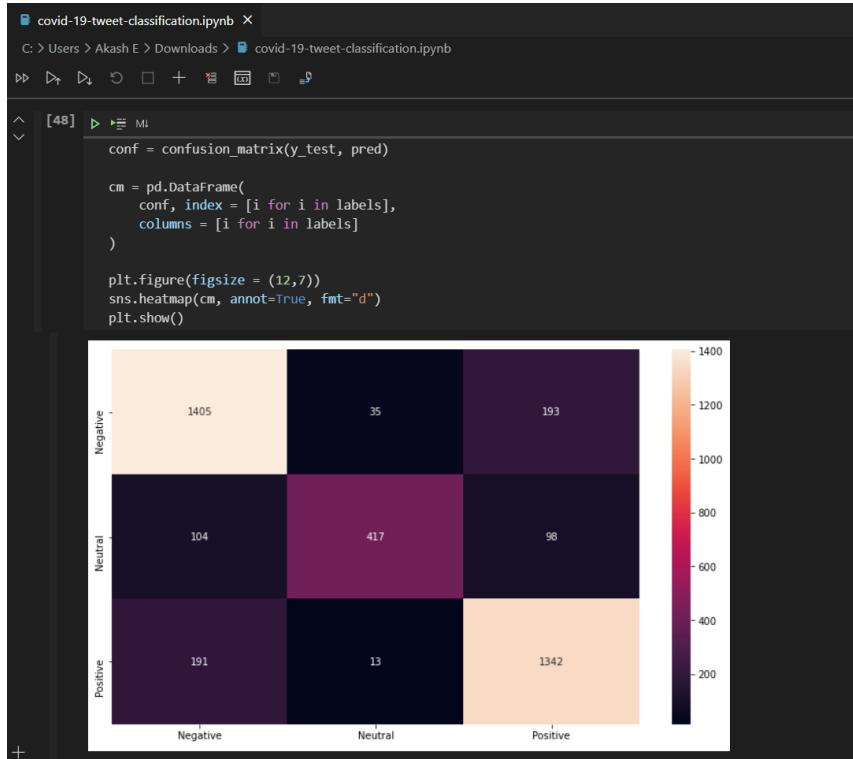
[47] ▶ ML
loss, acc = model.evaluate(x_test,y_test,verbose=0)
print('Test loss: {}'.format(loss))
print('Test Accuracy: {}'.format(acc))

Test loss: 0.47204768657684326
Test Accuracy: 0.8330700397491455

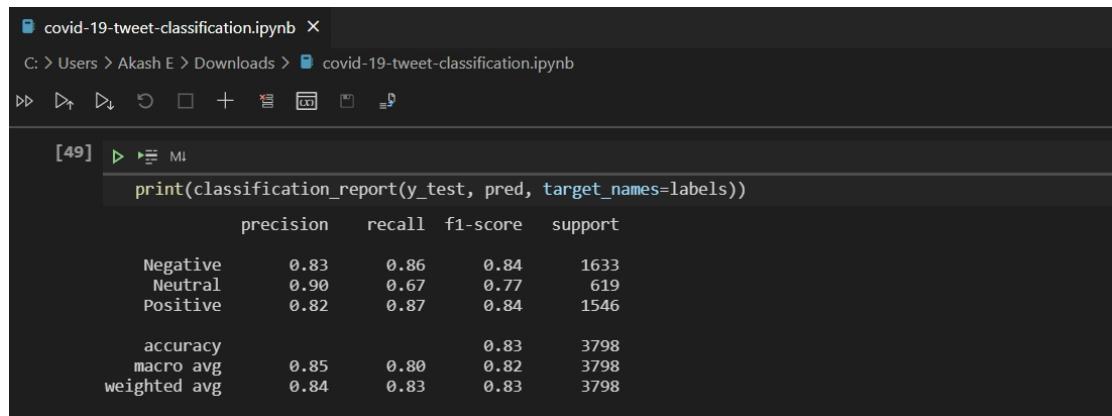
```

From the above code we can see that our model has a **testing accuracy of 83%** which is pretty good.

## Confusion Matrix :



## Classification Report :



The screenshot shows a Jupyter Notebook cell with the following code and output:

```
[49]: print(classification_report(y_test, pred, target_names=labels))
```

	precision	recall	f1-score	support
Negative	0.83	0.86	0.84	1633
Neutral	0.90	0.67	0.77	619
Positive	0.82	0.87	0.84	1546
accuracy			0.83	3798
macro avg	0.85	0.80	0.82	3798
weighted avg	0.84	0.83	0.83	3798

Training Accuracy	87%
Validation Accuracy	84%
Testing Accuracy	83%

## 6. CONCLUSION :

Hence we used the Bidirectional LSTM to perform sentiment analysis on the Covid-19 twitter dataset. We also got good training and testing accuracies while doing so.