CS6005

Deep Learning

Mini-Project Assignment - 3

Sentiment Analysis of Coronavirus Tweets

Steven F. Gilbert

2018103071

'P' Batch

11.05.21

Introduction

This Mini project aims to implement NLP - Natural Language Processing on a suitable dataset. NLP is a subfield of linguistics and artificial intelligence, in particular how to analyze and process large amounts of natural language data. Here we take a dataset consisting of tweets posted during the pandemic and we try to train the model based on the sentiments labelled and try to predict the sentiments of unseen tweets. Natural Language Processing or NLP is a field of Artificial Intelligence that gives the machines the ability to read, understand and derive meaning from human languages.

Problem Statement

The problem taken here is to analyze sentiments given in the dataset consisting of tweets posted during the pandemic. Here the tweets are labelled as 'Negative, Neutral and Positive'. Here we implement our model using an LSTM. LSTM helps to retain memory and does not suffer from vanishing gradients. We also use the nltk library for preprocessing the data such as to remove unwanted data and to preserve only the words that will help to train the model. The aim here is to see how well we can train the model and see how well it predicts the sentiments of unseen tweets.

Dataset Details

This dataset contains tweets posted during the Coronavirus pandemic. The tweets have been pulled from Twitter and manual tagging has been done to them. For training the tweets have been labelled with sentiments. The sentiments are 'Negative, Neutral and Positive'. There are about 41,000 tweets which will be split for training and testing purposes. All tweets are pulled from Twitter as is, so data preprocessing will have to be done.[1]

Sample Tweets:

OriginalTweet	Sentiment
TRENDING: New Yorkers encounter empty supermarket shelves (pictured, Wegmans in Brooklyn), sold-out	Negative
When I couldn't find hand sanitizer at Fred Meyer, I turned to #Amazon. But $114.97 for a 2 pack of Purell?	Positive
Find out how you can protect yourself and loved ones from #coronavirus. ?	Positive
#Election2020 #CDC https://t.co/29isZOewxu	Negative
#toiletpaper #dunnypaper #coronavirus #coronavirusaustralia #CoronaVirusUpdate #Covid_19 #9News #C	Neutral
Do you remember the last time you paid $2.99 a gallon for regular gas in Los Angeles?Prices at the pump a	Neutral
Voting in the age of #coronavirus = hand sanitizer ? #SuperTuesday https://t.co/z0BeL4O6Dk	Positive
cost twice as much"-@DrTedros #coronavirus	Neutral
HI TWITTER! I am a pharmacist. I sell hand sanitizer for a living! Or I do when any exists. Like masks, it is so	Negative
Anyone been in a supermarket over the last few days? Went to do my NORMAL shop last night & ??is	Positive
#DJSBU https://t.co/HhDJhyQ2Dc	Positive
https://t.co/2Ikkmimj4f https://t.co/RB9rtt7Nkc	Negative
https://t.co/aYQtLLGW1m	Negative
#Covid_19 Went to the Grocery Store, turns out all cleaning supplies have been bought out for fear of Coronavirus. My daughter's substitute teacher showed her class how to make hand sanitizer. Now I just need to buy some 100% alcohol. Wish me luck that any stores will have it.	Positive
While we were busy watching election returns and bracing for a Covid-19 outbreak, Trump nominated a ch	Positive
@HardeepSPuri @MoCA_GoI	Negative
https://t.co/7Zo2vqSPzY	Positive

Modules

The project code can be split into multiple modules, each with a specific function.

1) Reading the dataset

The data is first read from the csv file using pandas and put into a dataframe. The total no.of instances are about 41,000 tweets. This data will be split into training and testing at 40% and 30% split. The data here is raw directly from Twitter, so preprocessing will have to be done.

2) Preprocessing data

The tweets here contain unnecessary characters for the training like punctuations, stopwords, special characters and others. We use the nltk library to remove the stopwords. We also perform lemmatization on the words as well. We then use Keras' Tokenizer to convert each word into a number to feed it into the model for training. The target labels here are one-hot encoded.

3) Model Creation

Our model here consists first of an Embedding layer to which the inputs are fed. The embedding layer will create a dense vector of integers from the input data. Next for the analysis, we use LSTM - Long Short Term Memory, an RNN. The LSTM performs the analysis with dropout. A Global Max Pooling is added next and a flatten layer after that. We have 2 Dense layers of 32 units each having a 'relu' activation function and dropout of 0.4. The final layer is the output layer of 3 nodes, for the 3 classes and 'softmax' activation function.

Libraries

The main reason why python is very popular in the field of deep learning is that it boasts a multitude of libraries to perform several important and painstaking modules and functions.

Some of the libraries used in this project are as follows:

1) Tensorflow - TensorFlow is an end-to-end open source platform for machine learning .

2) Keras - It offers consistent & simple APIs, it minimizes the number of user actions required for common use cases .

3) Numpy - Package which provides data computational functions .

4) Matplotlib - Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python .

5) NLTK - The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing for English

Model Summary

To analyze the sentiments we use an embedding layer and an lstm with 2 other fully connected layers. A batch size of 256 is used for training the dataset. The model is trained for 10 epochs with early stop monitoring validation loss.

VGG16 architecture components :

1) **Embedding Layer :**
    a) Turns positive integers (indexes) into dense vectors of fixed size.
2) **LSTM Layer :**
    a) Overcomes Vanishing gradient problem.
    b) LSTM has a special architecture which enables it to forget the unnecessary information.
3) **Global Max Pooling Layer.**
    a) Downsamples the input representation by taking the maximum value over the time dimension.
    b) No parameter required so overfitting is avoided.
4) **Dropout Layer :**
    a) Dropout is a technique used to prevent the model from overfitting.
    b) It works by randomly setting the outgoing edges of hidden units to 0 at each update of the phase.

5) **Dense Layer :**

   a) The dense layer performs functions on the input and produces an output vector.

   b) It represents the layer which performs matrix vector multiplication.

<u>Model Architecture</u> :

Parameter	Value
Epochs	10 (early stopped at 7)
Optimizer	Adam
Learning Rate	0.001
Loss function	Categorical Cross Entropy

Dense Layer:

a) Units = 32, 32

b) Dropout = 0.4

c) Activation function = relu

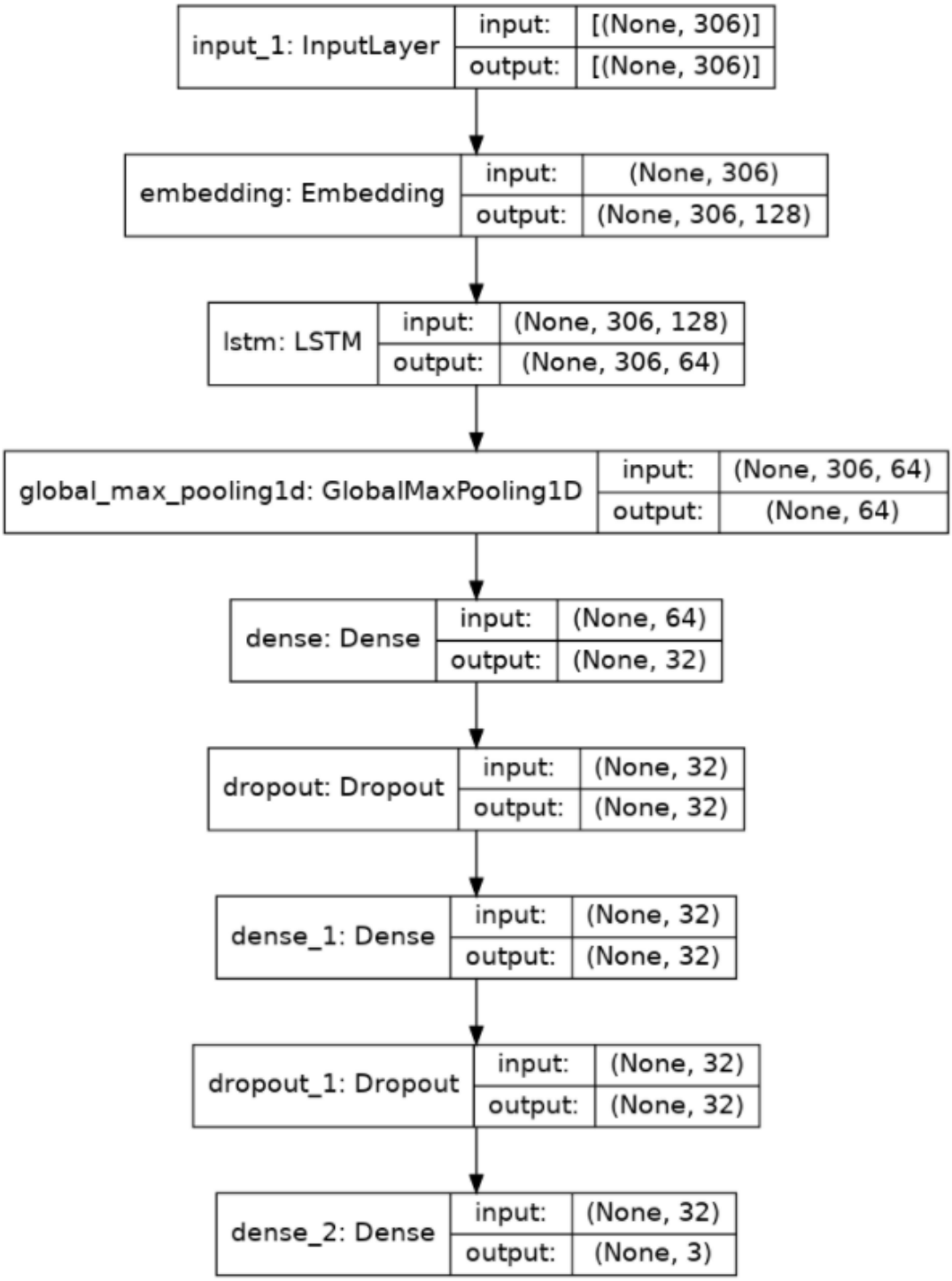
Output Layer:

a) Nodes = 3

b) Activation function = softmax

Model Summary

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 306)]             0
_________________________________________________________________
embedding (Embedding)        (None, 306, 128)          10834688
_________________________________________________________________
lstm (LSTM)                  (None, 306, 64)           49408
_________________________________________________________________
global_max_pooling1d (Global (None, 64)                0
_________________________________________________________________
dense (Dense)                (None, 32)                2080
_________________________________________________________________
dropout (Dropout)            (None, 32)                0
_________________________________________________________________
dense_1 (Dense)              (None, 32)                1056
_________________________________________________________________
dropout_1 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 99
=================================================================
Total params: 10,887,331
Trainable params: 10,887,331
Non-trainable params: 0
_________________________________________________________________
None
```


input_1: InputLayer
input: [(None, 306)]
output: [(None, 306)]
embedding: Embedding
input: (None, 306)
output: (None, 306, 128)
lstm: LSTM
input: (None, 306, 128)
output: (None, 306, 64)
global_max_pooling1d: GlobalMaxPooling1D
input: (None, 306, 64)
output: (None, 64)
dense: Dense
input: (None, 64)
output: (None, 32)
dropout: Dropout
input: (None, 32)
output: (None, 32)
dense_1: Dense
input: (None, 32)
output: (None, 32)
dropout_1: Dropout
input: (None, 32)
output: (None, 32)
dense_2: Dense
input: (None, 32)
output: (None, 3)

Coding Snapshots

1) Importing Packages

```
import numpy as np
import pandas as pd
from collections import Counter
from sklearn.preprocessing import LabelEncoder
```

```
from tensorflow.keras.models import Sequential
from keras.models import Model
from keras.layers.embeddings import Embedding
from keras.layers.merge import Concatenate
from tensorflow.keras import regularizers
from tensorflow.keras.layers import Dense,Dot,Conv1D,MaxPooling1D,Activation,Dro
pout,LSTM,Flatten,GlobalMaxPool1D,Input,BatchNormalization,Bidirectional
```

2) Reading the dataset into dataframe

```
import pandas as pd
df = pd.read_csv("../input/covid-19-nlp-text-classification/Corona_NLP_train.cs
v",encoding='latin1')
```

	UserName	ScreenName	Location	TweetAt	OriginalTweet	Sentiment
0	3799	48751	London	16-03-2020	@MeNyrbie @Phil_Gahan @Chrisitv https://t.co/i...	Neutral
1	3800	48752	UK	16-03-2020	advice Talk to your neighbours family to excha...	Positive
2	3801	48753	Vagabonds	16-03-2020	Coronavirus Australia: Woolworths to give elde...	Positive
3	3802	48754	NaN	16-03-2020	My food stock is not the only one which is emp...	Positive

3) Label Encoding

```
from sklearn.preprocessing import LabelBinarizer
lb = LabelBinarizer()
y = lb.fit_transform(df.Sentiment)
```

```
lb.classes_
```

```
array(['Negative', 'Neutral', 'Positive'], dtype='<U8')
```

4) Removing special characters

```
df["OriginalTweet"]=df["OriginalTweet"].str.replace('[^\w\s]', '')
df.head()
```

5) Removing Stopwords

```
import nltk
from nltk.corpus import stopwords
sw=stopwords.words('english')
df["OriginalTweet"]=df["OriginalTweet"].apply(lambda x: " ".join(x for x in x.sp
lit() if x not in sw))
df.head()
```

6) Lemmatization

```
from textblob import Word, TextBlob
df["OriginalTweet"]=df["OriginalTweet"].apply(lambda x:" ".join([Word(word).lemm
atize() for word in x.split()]))
df.head()
```

7) Conversion to sequences

```
from keras.preprocessing import text, sequence
from keras.preprocessing.text import Tokenizer

max_length = max(lengths)
vocab_size = 10000
oov_token = "<OOV>"
tokenizer = Tokenizer(num_words=vocab_size, oov_token=oov_token)
tokenizer.fit_on_texts(seqs)
#print(tokenizer.word_counts)
#print(tokenizer.document_count)
#print(tokenizer.word_index)
#print(tokenizer.word_docs)
```

```
X_seq = tokenizer.texts_to_sequences(seqs)
X_seq = sequence.pad_sequences(X_seq, maxlen=max_length)
```

8) Model creation

```
input1=Input(shape=(max_length,))
e = Embedding(len(tokenizer.word_index)+1,128)(input1)
#lstm1=LSTM(32,return_sequences=True,dropout=0.5)(e)
lstm2=LSTM(64,return_sequences=True,dropout=0.5)(e)
gp1=GlobalMaxPool1D()(lstm2)
f1 = Flatten()(gp1)
d1 = Dense(32, activation='relu',kernel_regularizer='l2')(gp1)
dr1= Dropout(0.4)(d1)
d2 = Dense(32, activation='relu',kernel_regularizer='l2')(dr1)
dr2= Dropout(0.4)(d2)
```

```
output = Dense(no_classes, activation='softmax')(dr2)
model = Model(inputs=input1, outputs=output)
```

9) Model compilation

```
from tensorflow.keras.optimizers import Adam
opt = Adam(learning_rate=0.001)
model.compile(loss='categorical_crossentropy', optimizer='Adam', metrics=['ac
c'])
```

10) Train Test split

```
from sklearn.model_selection import train_test_split
X_seq = np.array(X_seq)
X_train,X_test,y_train,y_test= train_test_split(X_seq,y,train_size=0.4,test_size
=0.3,shuffle=True)
```

11) Training

```
history = model.fit(x=X_train, y=y_train, batch_size=256, epochs=10,validation_s
plit=0.3,shuffle=True,callbacks=[early_stop])
```

12) Evaluation

```
print("Evaluate on test df")
results = model.evaluate(X_test, y_test, batch_size=128)
```

Results

1) Sequence Conversion

```
[[   0    0    0 ...    1    1    1]
 [   0    0    0 ... 1011 2678  118]
 [   0    0    0 ...    3   80    1]
 ...
 [   0    0    0 ...    1   30   34]
 [   0    0    0 ...    2    3    2]
 [   0    0    0 ...    3  777    1]]
```

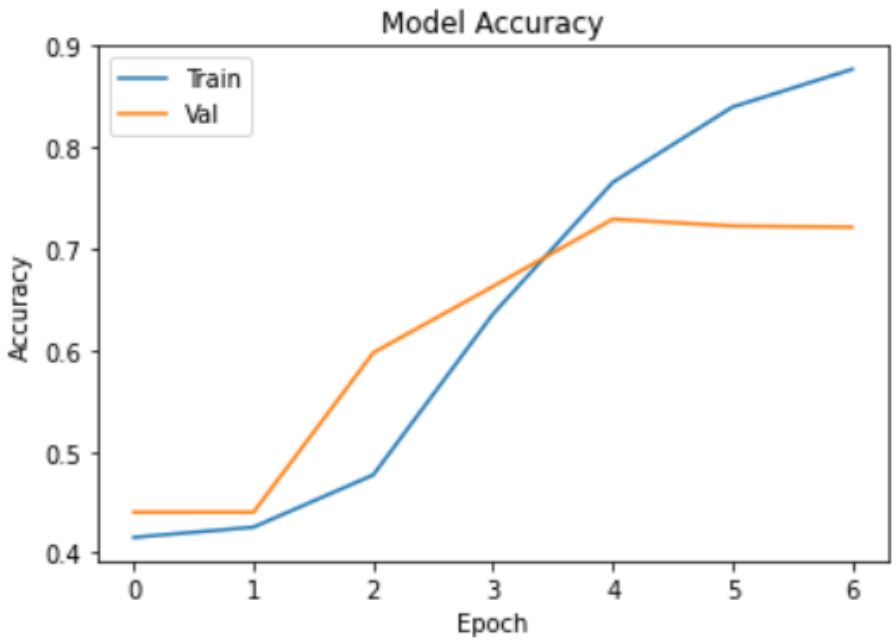
2) Training Model (early stopped at 7)

```
Epoch 1/10
46/46 [==============================] - 13s 227ms/step - loss: 1.7436 - acc: 0.4160 - v
al_loss: 1.5006 - val_acc: 0.4402
Epoch 2/10
46/46 [==============================] - 10s 212ms/step - loss: 1.4476 - acc: 0.4243 - v
al_loss: 1.2496 - val_acc: 0.4402
Epoch 3/10
46/46 [==============================] - 9s 204ms/step - loss: 1.2147 - acc: 0.4448 - va
l_loss: 1.0390 - val_acc: 0.5975
Epoch 4/10
46/46 [==============================] - 9s 205ms/step - loss: 0.9704 - acc: 0.6140 - va
l_loss: 0.8774 - val_acc: 0.6629
Epoch 5/10
46/46 [==============================] - 9s 201ms/step - loss: 0.7473 - acc: 0.7471 - va
l_loss: 0.7971 - val_acc: 0.7293
Epoch 6/10
46/46 [==============================] - 10s 210ms/step - loss: 0.6359 - acc: 0.8330 - v
al_loss: 0.8584 - val_acc: 0.7226
Epoch 7/10
46/46 [==============================] - 9s 198ms/step - loss: 0.5328 - acc: 0.8697 - va
l_loss: 0.8765 - val_acc: 0.7214
```

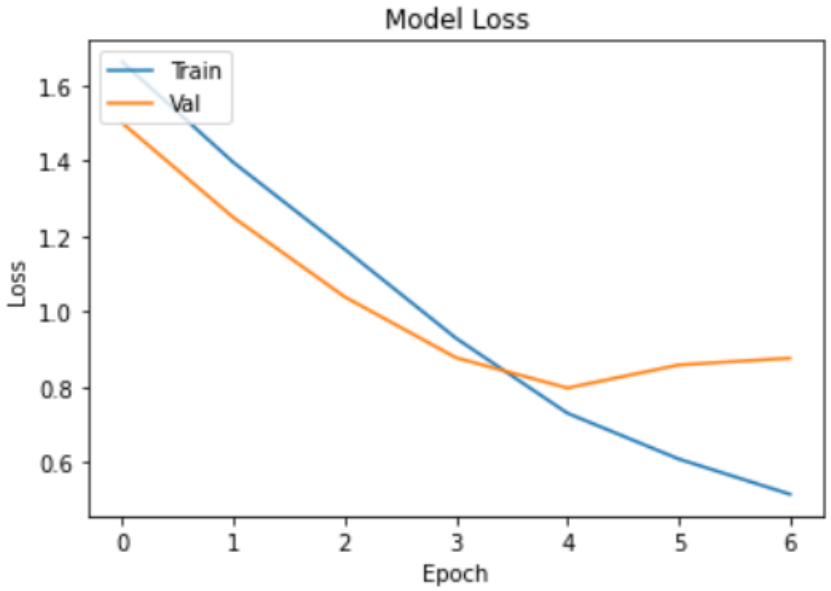
3) Final Testing Accuracy

```
Test Loss 0.8115623593330383
Test Accuracy: 0.7300777435302734
```

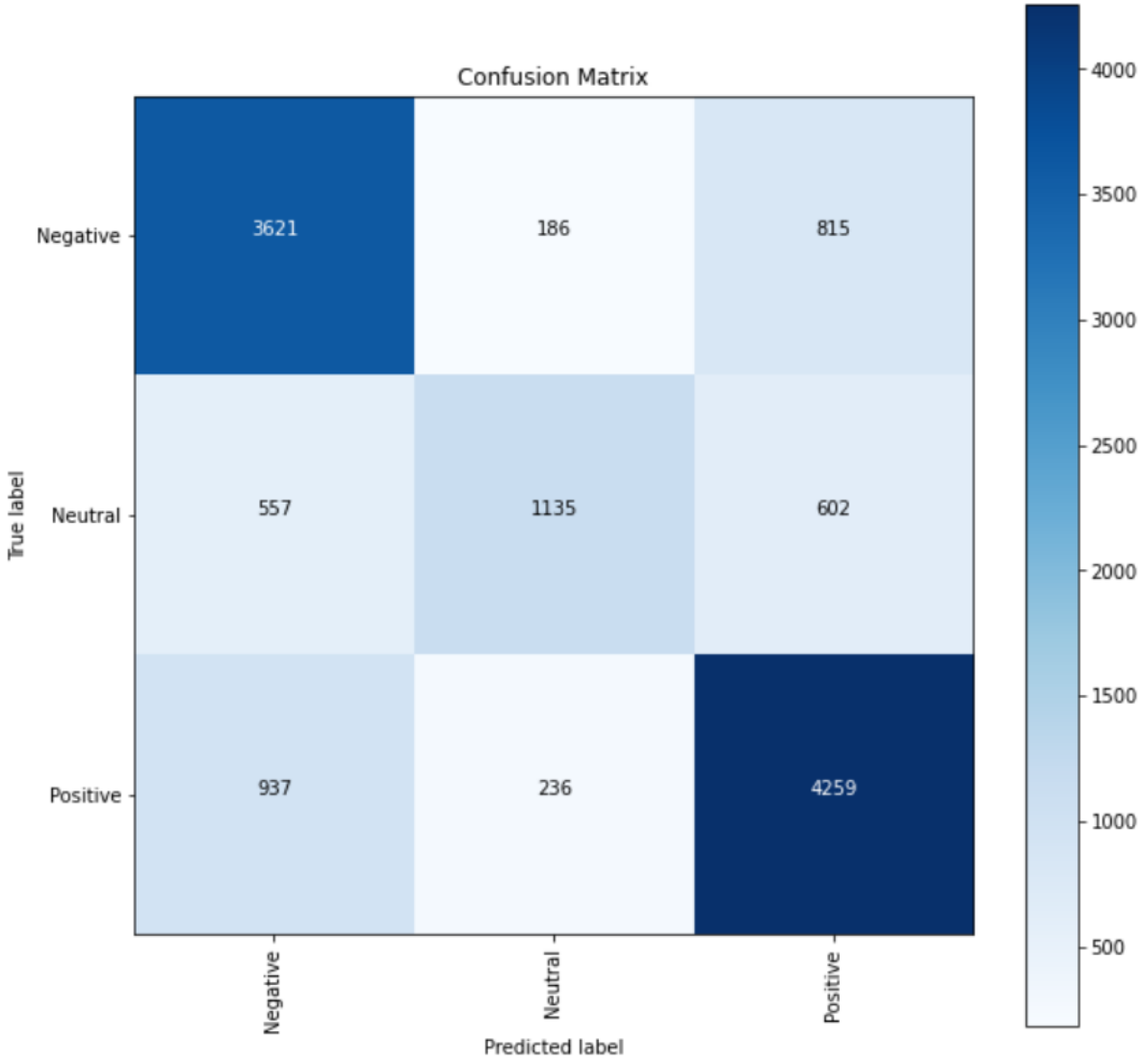
4) Learning Curve



5) Loss curve



6) Confusion Matrix



7) Classification Report

```
Classification Report
              precision    recall  f1-score   support

           0       0.71      0.78      0.74      4622
           1       0.73      0.49      0.59      2294
           2       0.75      0.78      0.77      5432

    accuracy                           0.73     12348
   macro avg       0.73      0.69      0.70     12348
weighted avg       0.73      0.73      0.73     12348
```

Conclusion

As seen from the results, we have been able to analyze the tweets for different sentiments like Negative, Neutral, Positive. The final Testing accuracy was obtained to be 73% with a loss value of 0.8115.

A function has been created to input any tweet to test this model.

```
predict_emotion("Global stocks plummeted today due to fear surrounding #COVID2019 and fallin
g oil prices, which saw the biggest one-day drop since the 1991 Gulf War")
```

```
['Negative']
```

The concepts of NLP have been clearly understood and implemented with results being derived.

References

[1] https://www.kaggle.com/datatattle/covid-19-nlp-text-classification