COMPARISON OF Bi-LSTM

WITH INDEX | WORD2VEC | GLOVE EMBEDDING

BERT

Sachith M. Gunawardane

Table of Contents

1 Summary 3

2 Exploratory Data Analysis (EDA) 4

2.1 Data duplication 4

2.2 Data distribution 5

2.3 Data Cleanup and Visualisation 6

2.4 Training and Validation Data Split 7

3 Bi-LSTM Model with Index Embedding 7

4 Word2Vec Embedding 9

5 GloVe Embedding 10

6 Transformers - BERT 11

7 TensorBoard 12

Dataset:

The dataset contains tweets from individuals discussing their health conditions or general health topics. These tweets are labelled personal health mentions (1) or non-personal health mentions (0).

Objective:

1. Create a sequence / RNN model (LSTM or Bi-LSTM) for classifying tweets.
2. Performance comparison for two or more models.

Methodology:

1. I engineered an efficient Bi-LSTM model incorporating index embedding through Keras tuner for the purpose of tweet classification.
2. Comprehensive comparisons were conducted, exploring various embedding techniques, including Index, Word2Doc, GloVe, and BERT. This encompasses transfer learning with GloVe and utilising Transformers with BERT.

The document is organised into the following sections: summary, exploratory data analysis (EDA), Bi-LSTM with Index, Word2Vec Embedding, GloVe Embedding, and BERT Transformer.

1 Summary

The performance of various models in the context of "***Attention Is All You Need***" can be summarised by highlighting that the BERT model outperformed all others. Specifically, BERT achieved notable **validation and test accuracies of 0.9169 and 0.8645**, respectively. Additionally, the ROC curve demonstrated a high AUC value of 0.923, underscoring the model's robustness.

GloVe (utilising transfer learning) followed closely as the runner-up and exhibited a respectable performance with an AUC value of 0.892. The corresponding validation and test accuracies were 0.8934 and 0.8240, respectively.

The Bi-LSTM model, incorporating index embedding, performed well, securing a commendable AUC of 0.844. Its validation and test accuracies stood at 0.9040 and 0.8010, respectively.

Surprisingly, the performance of the Bi-LSTM model with Word2Vec embedding lagged behind, revealing an AUC of 0.779. The model's validation and test accuracies were 0.8003 and 0.6597, respectively. Despite expectations for Word2Vec to outperform index embedding, the results indicate otherwise.

In light of these findings, it can be concluded that, given the available dataset, the current model configuration with BERT stands out as the most effective. However, the potential for further improvement exists with the inclusion of more data, as it not only enhances the accuracy of models but also allows for the exploration of more sophisticated model architectures.

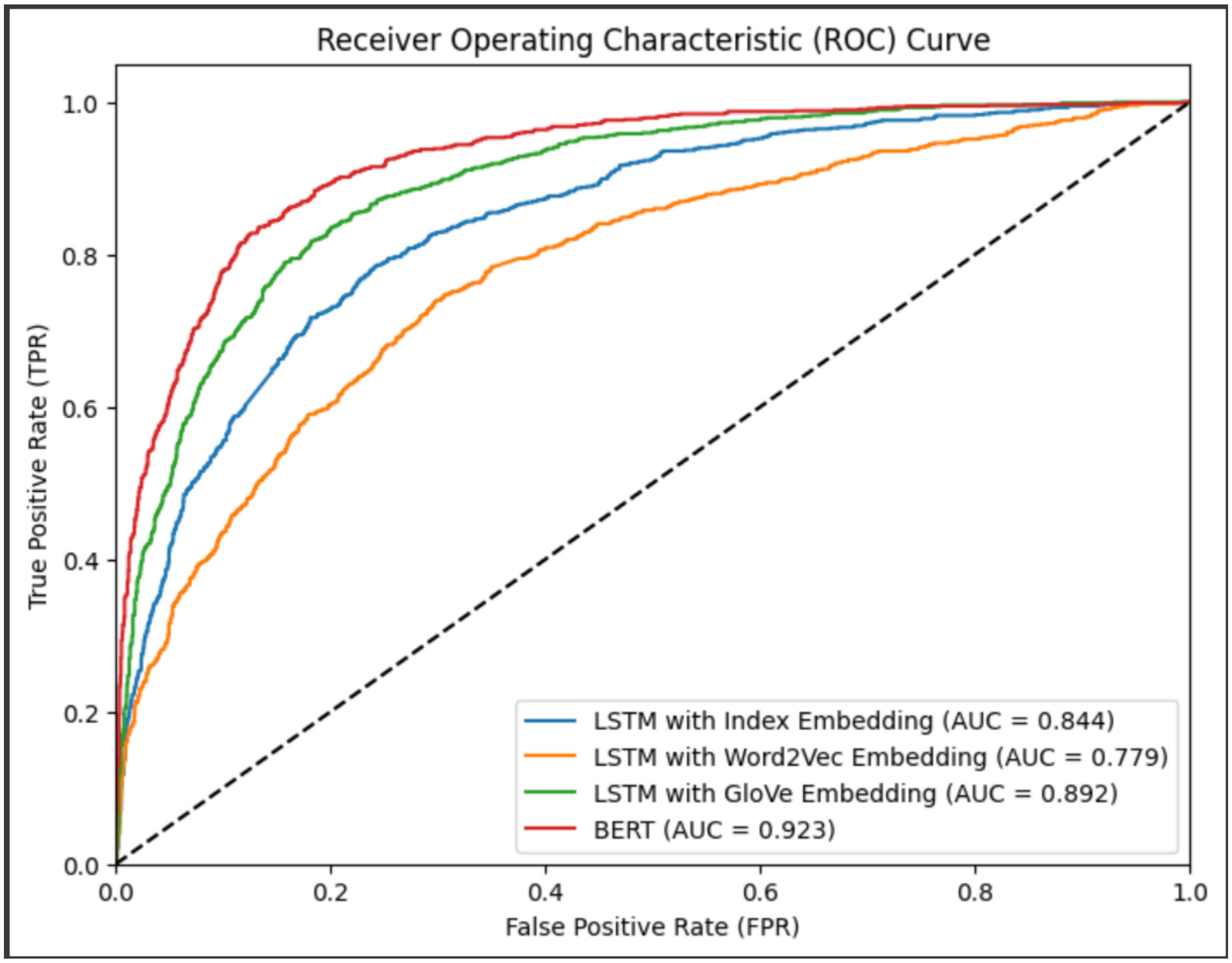


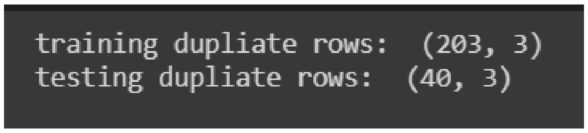
Figure 1-1 - ROC Curve

Refer .ipynb to more details and explanation for each step.

2 Exploratory Data Analysis (EDA)

2.1 Data duplication

The training dataset comprises 9,991 records, while the test dataset comprises 3,331 records. Both datasets initially contained duplicate data entries. However, a decision was made to eliminate duplicate data at this point. This choice was influenced by the observation that the majority of duplicate entries belonged to the non-personal health mentions category. Given the dataset's imbalance, it was deemed necessary to perform data upsampling before proceeding to the training stage. After the removal of duplicates, the total number of records in the training set was reduced to 9,788 and in the testing set to 3,291.

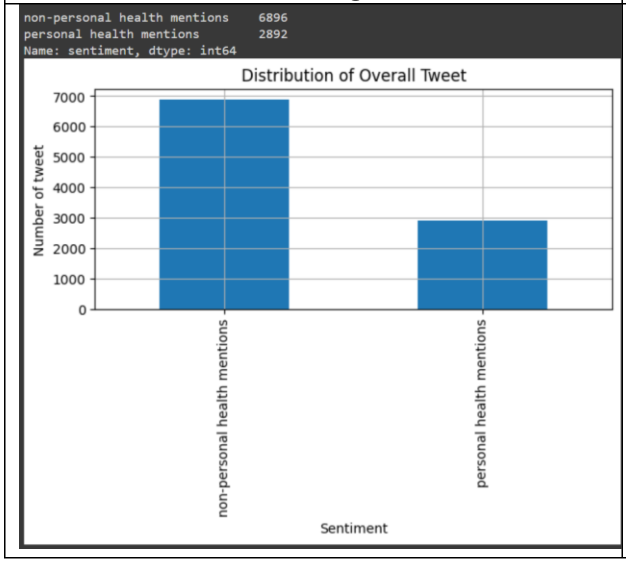
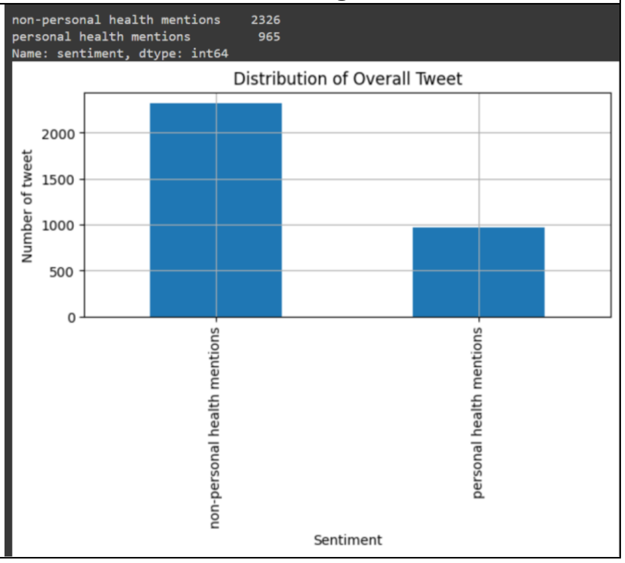


```
training dupliate rows:  (203, 3)
testing dupliate rows:  (40, 3)
```

Figure 2-1 - Duplicate rows

2.2 Data distribution

There are no null values. Data distribution between two groups as follows.

Training	Testing
	

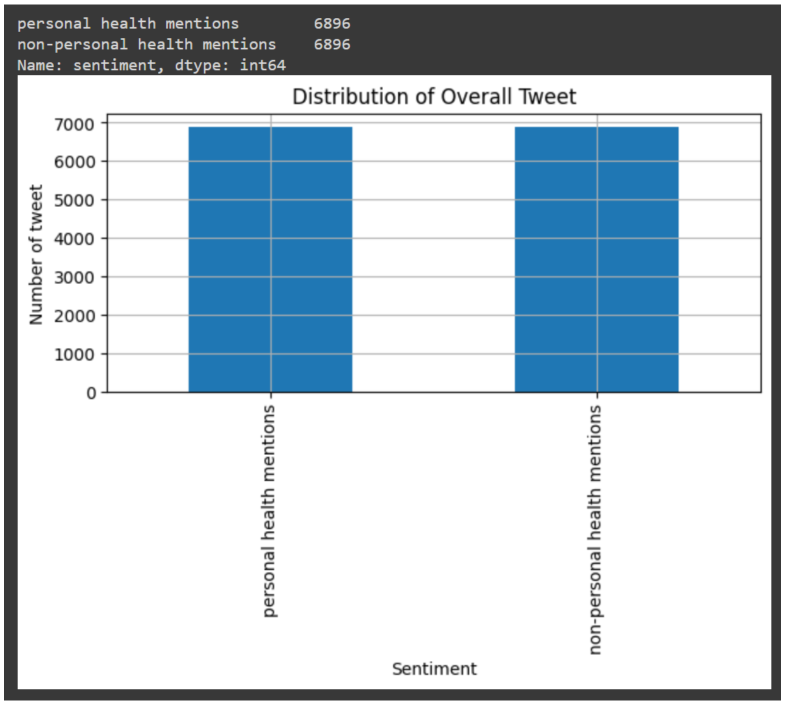
Training:

```
non-personal health mentions    6896
personal health mentions        2892
Name: sentiment, dtype: int64
```

Testing:

```
non-personal health mentions    2326
personal health mentions         965
Name: sentiment, dtype: int64
```

Given the imbalance in the dataset mentioned earlier, the decision was made to upsample the personal health mentions training data to match the size of the non-personal health mentions data. This upsampling process was executed after completing the data cleanup in the notebook.



```
personal health mentions        6896
non-personal health mentions    6896
Name: sentiment, dtype: int64
```

Figure 2-2 – Upsample Training Dataset

2.3 Data Cleanup and Visualisation

As part of the data cleanup, the following activities were performed,

1. Text to lowercase
2. Remove any html taags
3. Remove all punctuations
4. Remove all non-alpha characters
5. Remove all stopwords
6. Apply text Lemmatisation

Number words in each tweet as follows for each category in test datasets.

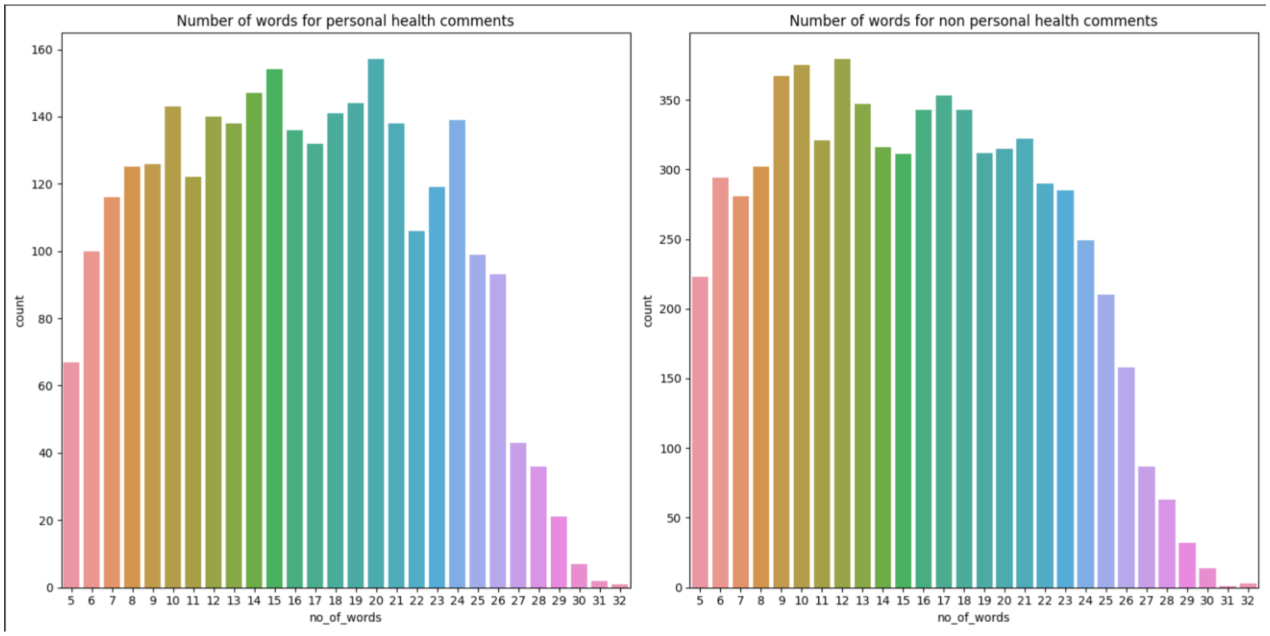


Figure 2-3 - No of words for category

Below is the word cloud for categories,

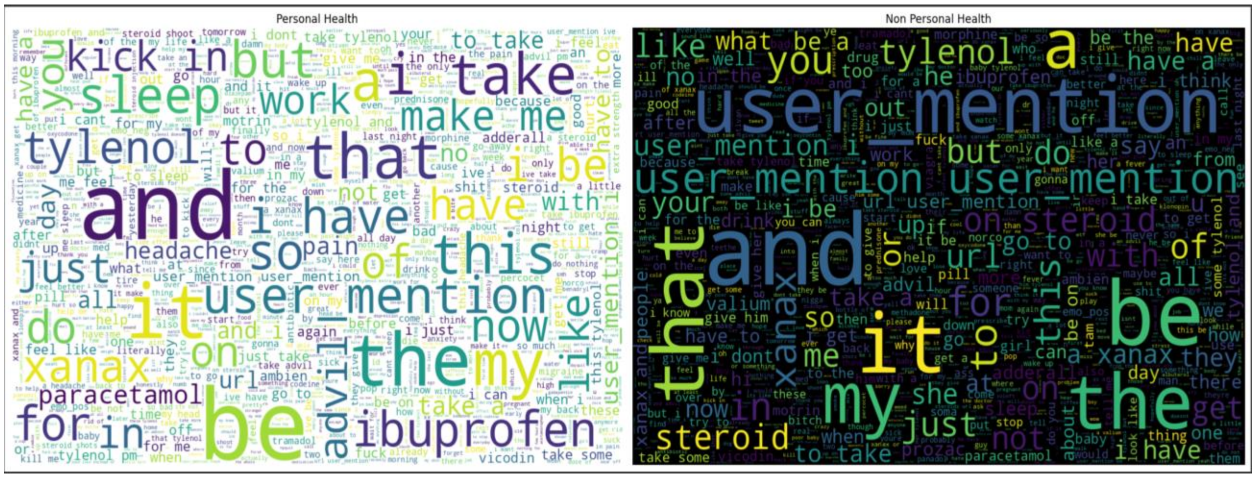
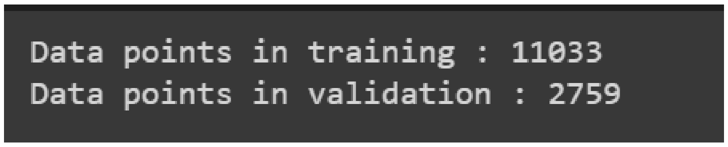


Figure 2-4 - Word Cloud

2.4 Training and Validation Data Split

The training data was subsequently divided into training and validation sets, with an 80% to 20% ratio.

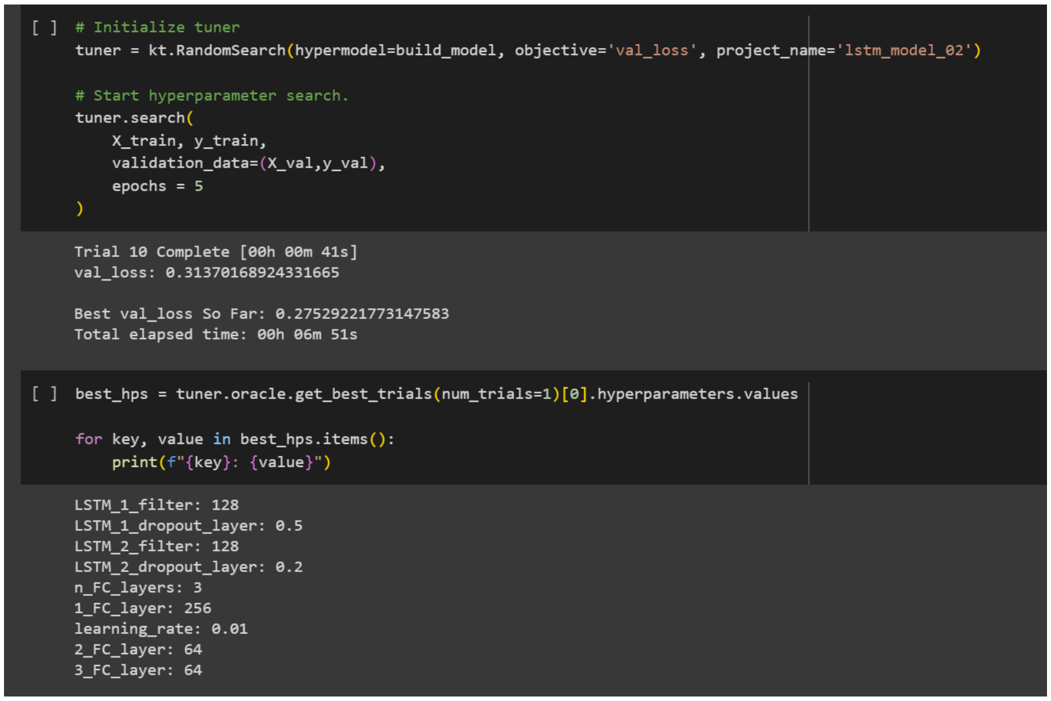


```
Data points in training : 11033
Data points in validation : 2759
```

Figure 2-5 - Validation split

3 Bi-LSTM Model with Index Embedding

The optimal model was crafted with the assistance of the Keras tuner, fine-tuning hyperparameters as outlined below:

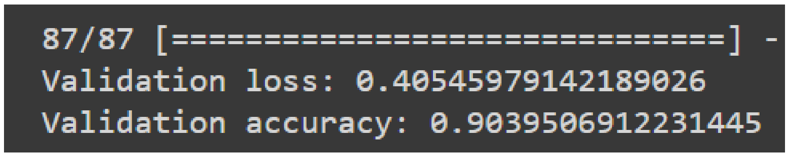


```
[ ]  # Initialize tuner
     tuner = kt.RandomSearch(hypermodel=build_model, objective='val_loss', project_name='lstm_model_02')

     # Start hyperparameter search.
     tuner.search(
         X_train, y_train,
         validation_data=(X_val,y_val),
         epochs = 5
     )

     Trial 10 Complete [00h 00m 41s]
     val_loss: 0.31370168924331665

     Best val_loss So Far: 0.27529221773147583
     Total elapsed time: 00h 06m 51s
```

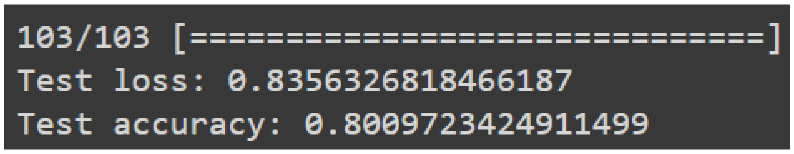
```
[ ]  best_hps = tuner.oracle.get_best_trials(num_trials=1)[0].hyperparameters.values

     for key, value in best_hps.items():
         print(f"{key}: {value}")

     LSTM_1_filter: 128
     LSTM_1_dropout_layer: 0.5
     LSTM_2_filter: 128
     LSTM_2_dropout_layer: 0.2
     n_FC_layers: 3
     1_FC_layer: 256
     learning_rate: 0.01
     2_FC_layer: 64
     3_FC_layer: 64
```

Figure 3-1 - Hyperparameter Tuning

Validation and Test accuracy of the model

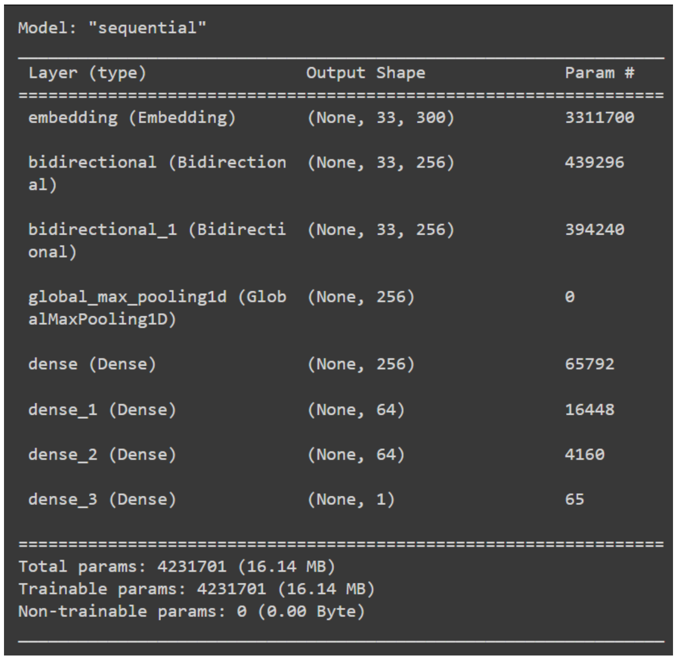


```
87/87 [==============================] -
Validation loss: 0.40545979142189026
Validation accuracy: 0.9039506912231445
```



```
103/103 [==============================]
Test loss: 0.8356326818466187
Test accuracy: 0.8009723424911499
```

Model architecture,



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 33, 300)           3311700

 bidirectional (Bidirection  (None, 33, 256)           439296
 al)

 bidirectional_1 (Bidirecti  (None, 33, 256)           394240
 onal)

 global_max_pooling1d (Glob  (None, 256)               0
 alMaxPooling1D)

 dense (Dense)               (None, 256)               65792

 dense_1 (Dense)             (None, 64)                16448

 dense_2 (Dense)             (None, 64)                4160

 dense_3 (Dense)             (None, 1)                 65

=================================================================
Total params: 4231701 (16.14 MB)
Trainable params: 4231701 (16.14 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 3-2 - Bi-LSTM - Index – Architecture

Loss and Accuracy

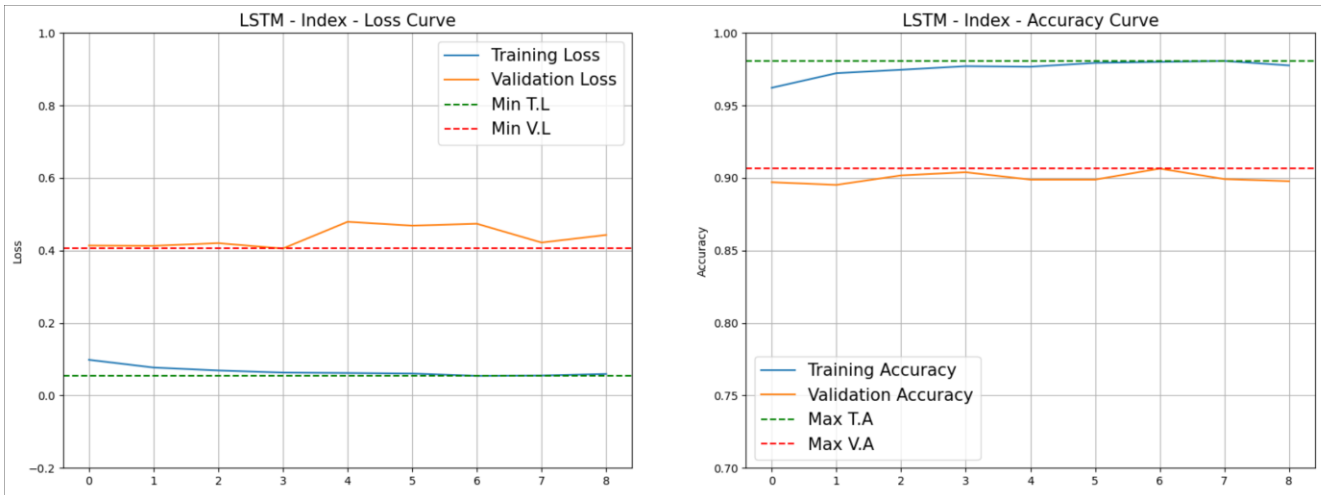
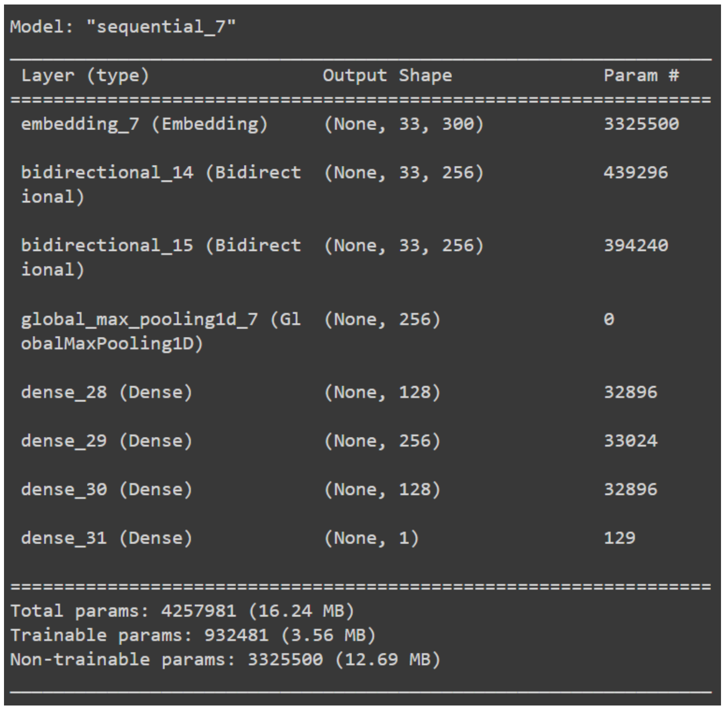


Figure 3-3 - Loss and Accuracy

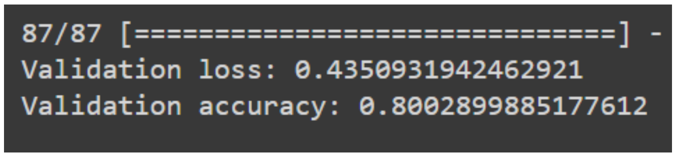
4 Word2Vec Embedding

Model architecture,



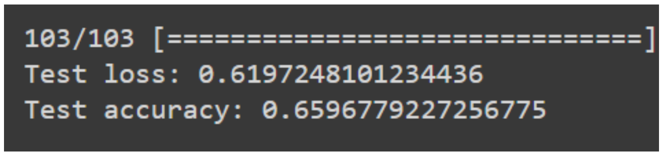
```
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_7 (Embedding)     (None, 33, 300)           3325500

 bidirectional_14 (Bidirect  (None, 33, 256)           439296
 ional)

 bidirectional_15 (Bidirect  (None, 33, 256)           394240
 ional)

 global_max_pooling1d_7 (Gl  (None, 256)               0
 obalMaxPooling1D)

 dense_28 (Dense)            (None, 128)               32896

 dense_29 (Dense)            (None, 256)               33024

 dense_30 (Dense)            (None, 128)               32896

 dense_31 (Dense)            (None, 1)                 129

=================================================================
Total params: 4257981 (16.24 MB)
Trainable params: 932481 (3.56 MB)
Non-trainable params: 3325500 (12.69 MB)
_________________________________________________________________
```

Validation Loss and Accuracy



```
87/87 [==============================] -
Validation loss: 0.4350931942462921
Validation accuracy: 0.8002899885177612
```

Test Loss and Accuracy



```
103/103 [==============================]
Test loss: 0.6197248101234436
Test accuracy: 0.6596779227256775
```

Graph for Loss and Accuracy,

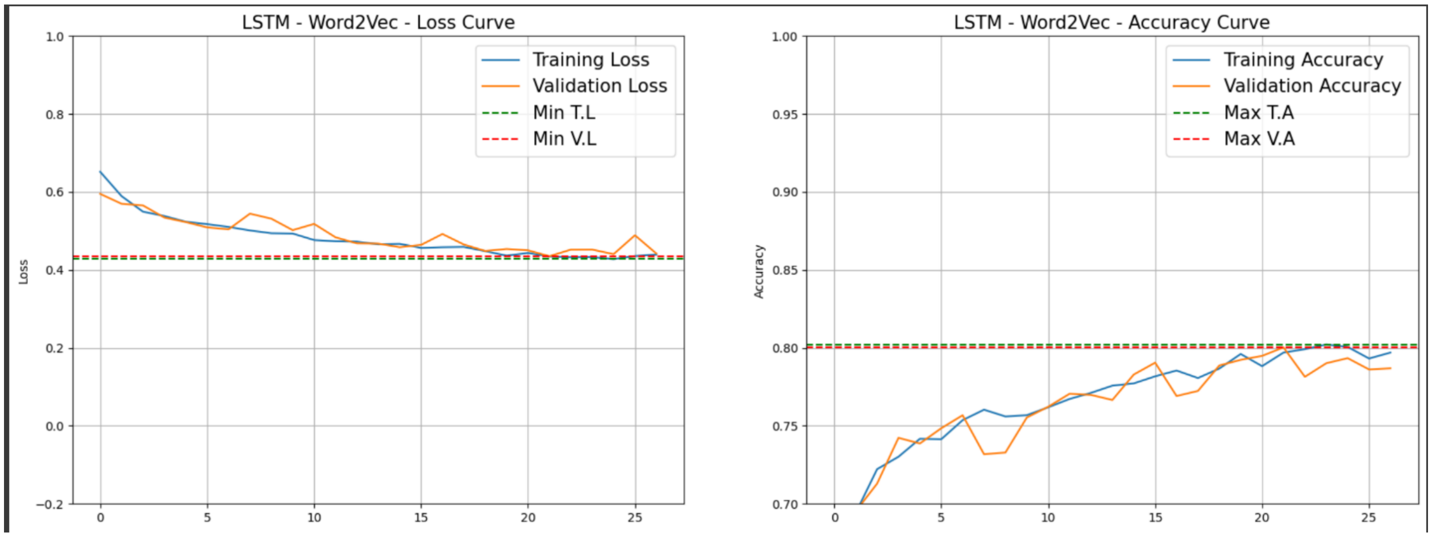
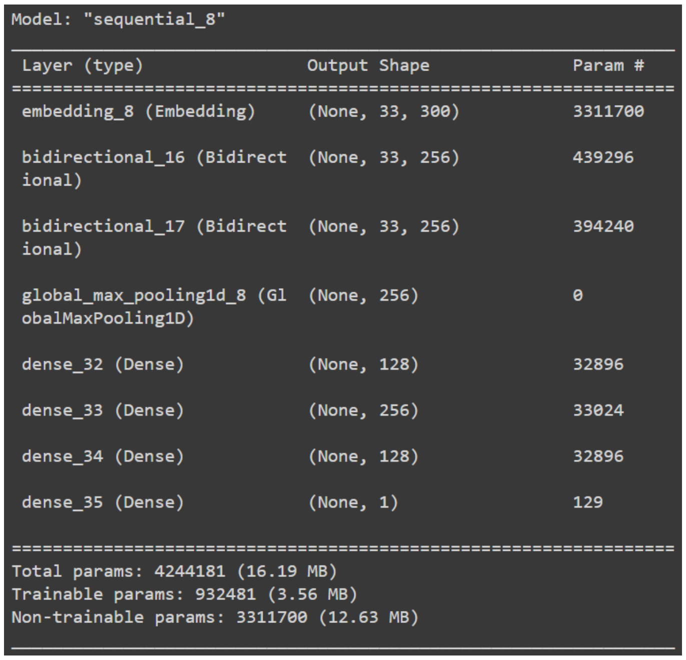


Figure 4-1 - Loss and Accuracy

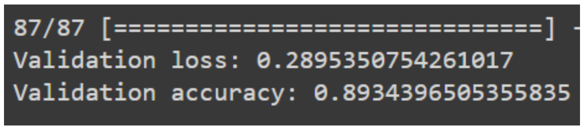
5 GloVe Embedding

Model Architecture,



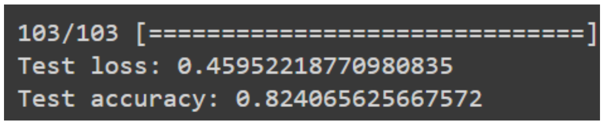
```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_8 (Embedding)     (None, 33, 300)           3311700

 bidirectional_16 (Bidirect  (None, 33, 256)           439296
 ional)

 bidirectional_17 (Bidirect  (None, 33, 256)           394240
 ional)

 global_max_pooling1d_8 (Gl  (None, 256)               0
 obalMaxPooling1D)

 dense_32 (Dense)            (None, 128)               32896

 dense_33 (Dense)            (None, 256)               33024

 dense_34 (Dense)            (None, 128)               32896

 dense_35 (Dense)            (None, 1)                 129

=================================================================
Total params: 4244181 (16.19 MB)
Trainable params: 932481 (3.56 MB)
Non-trainable params: 3311700 (12.63 MB)
_________________________________________________________________
```

Validation Loss and Accuracy



```
87/87 [==============================] -
Validation loss: 0.2895350754261017
Validation accuracy: 0.8934396505355835
```

Test Loss and Accuracy



```
103/103 [==============================]
Test loss: 0.45952218770980835
Test accuracy: 0.824065625667572
```

Graph for Loss and Accuracy,

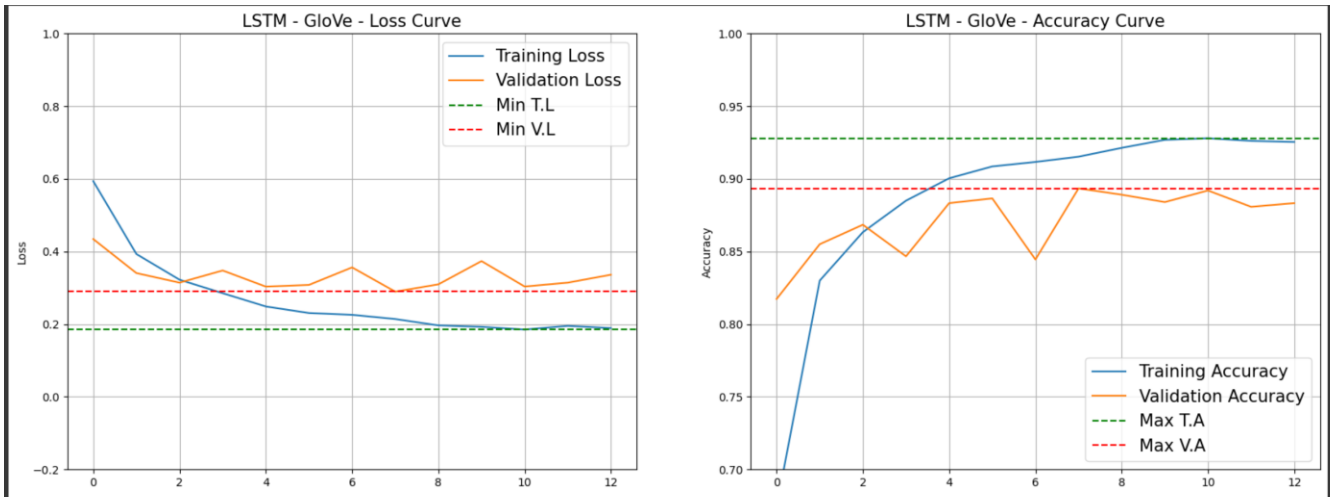
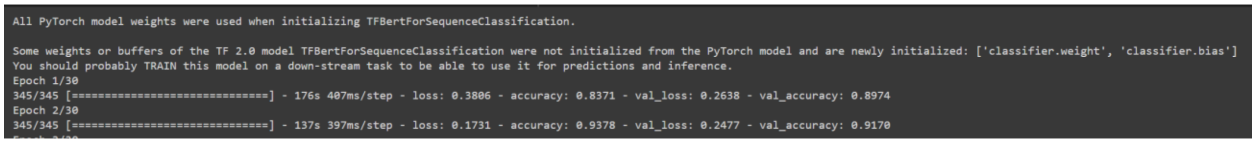


Figure 5-1 - Loss and Accuracy

6 Transformers - BERT

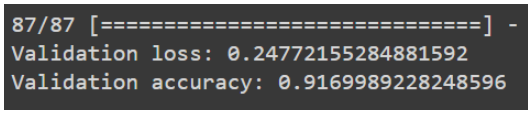
Bert training,



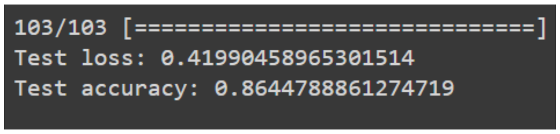
```
All PyTorch model weights were used when initializing TFBertForSequenceClassification.

Some weights or buffers of the TF 2.0 model TFBertForSequenceClassification were not initialized from the PyTorch model and are newly initialized: ['classifier.weight', 'classifier.bias']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
Epoch 1/30
345/345 [==============================] - 176s 407ms/step - loss: 0.3806 - accuracy: 0.8371 - val_loss: 0.2638 - val_accuracy: 0.8974
Epoch 2/30
345/345 [==============================] - 137s 397ms/step - loss: 0.1731 - accuracy: 0.9378 - val_loss: 0.2477 - val_accuracy: 0.9170
```

Validation Loss and Accuracy

Test Loss and Accuracy



```
87/87 [==============================] -
Validation loss: 0.24772155284881592
Validation accuracy: 0.9169989228248596
```



```
103/103 [==============================]
Test loss: 0.41990458965301514
Test accuracy: 0.8644788861274719
```

Graph for Loss and Accuracy,

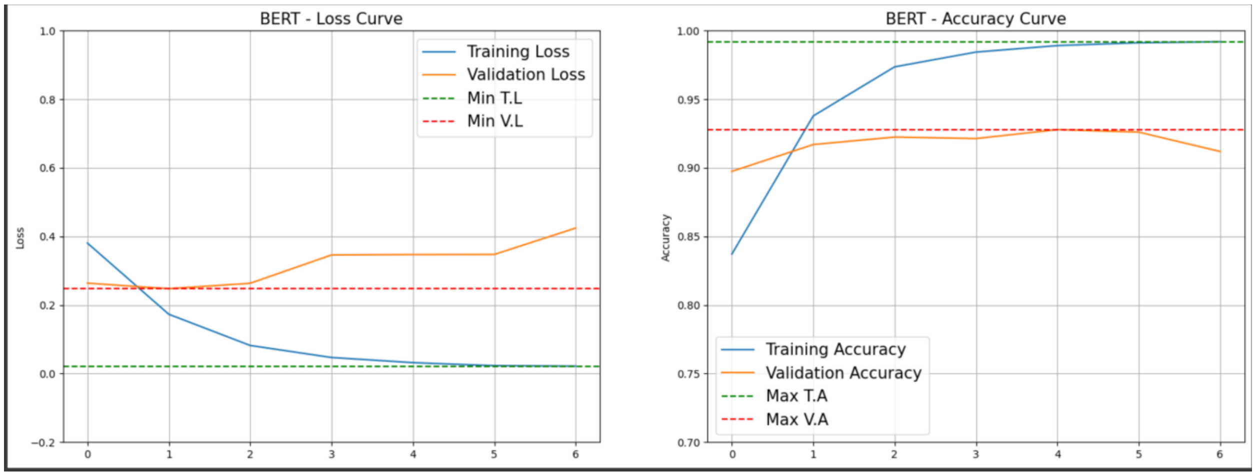
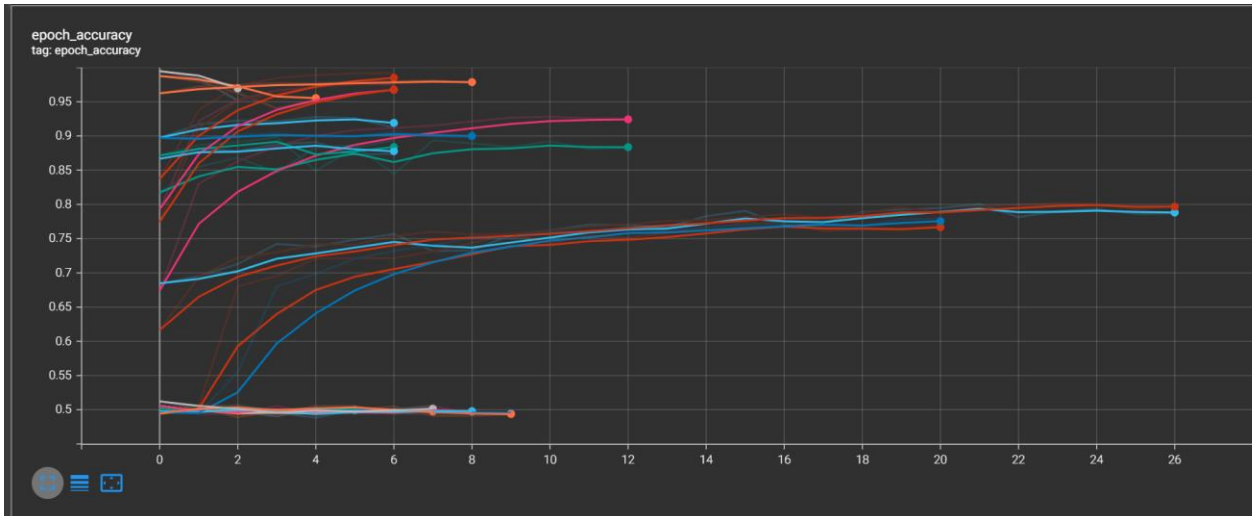
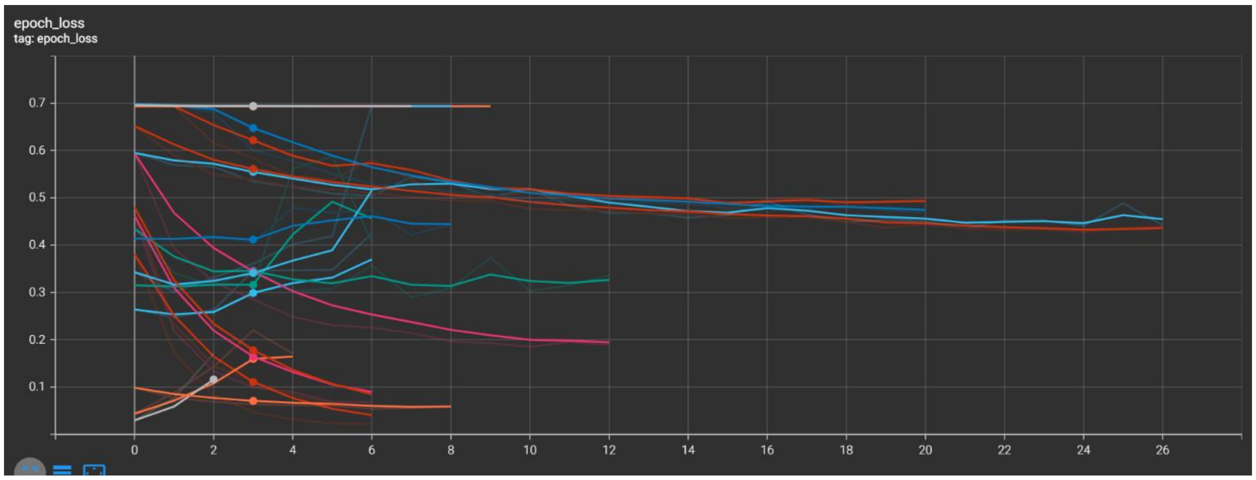


Figure 6-1 - Loss and Accuracy

7 TensorBoard


epoch_accuracy
tag: epoch_accuracy


epoch_loss
tag: epoch_loss