NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

Student's full-name: Spanos Nikolaos
Academic Number: 7115112100023
Teacher: Koumparakis Manolis

3rd academic homework exercise of course Artificial Intelligence II
Academic year: 2022-2023
Master of Science in Computer Science

Athens, January 2023

Table of Contents

`

Sentiment analysis on imdb movie reviews using Deep Learning and PyTorch

The code is thoroughly developed in the Google Colab notebook (Project_III.ipynb) included in the project deliverables. This report will provide a thorough description of the results and the steps followed to develop and evaluate different model classifiers for the sentiment analysis on imdb movie reviews.

1. Importing the dataset

The dataset used is the exact same movies reviews from HW1 & HW2. Thus, the transformation and preparation of the text sentences and the target sentiment for NLP experiments has already been developed in the previous Homework(s). On this homework the dataset is already cleaned and prepared for Natural Language Preprocessing[1].

2. Feed forward neural networks with RNN layers - development, training, and evaluation

In this unit, I present the results from different experiments using pretrained GloVe embeddings. The hyper-parameters of each experiment were monitored and selected from the Optuna[2] framework.
The hyper-parameters tuned are:

- LSTM/GRU cells hidden units. [64 - 128]
- Number of LSTM/GRU (stacked) layers. [1 - 3]
- LSTM/GRU cells drop-out probability. [0.15 - 0.30]
- Linear layer(s) hidden units. [16 - 64]
- Dropout rate probability in the case of Dropout layer. [*0.25 – 0.75*]
- Number of sequential linear/dense layers. [0 - 3]
- Learning rate. [*0.005 - 0.01*]
- Learning rate scheduler. [step = 10-20, gamma = 0.5-0.85]
- Gradient clipping coefficient. [1.5 - 5.5]
- Batch size of data Iterator objects [128 - 256]

[1] Lemmatization, punctation removal, stop words removal, abbreviation transformation, shuffling, stratification.
[2] Optuna - A hyperparameter optimization framework

The fixed hyper-parameters:

- Embeddings dimension [50]. No other values of embeddings were tested.
- Bi-directional RNN models.
- Activation function [*ReLU*] & Activation function for output layer [*Sigmoid*].
- Loss function [*Binary Cross Entropy*].
- Optimizer class [*Adam*].
- Epochs [50 to 80].
- Sentence length (padding) [165 words].

Optuna works with trials. Each trial is a different combination of hyper-parameter values. The number of trials used is 8. Thus, 8 models have been trained on each advanced complexity experiment. The experiments marked as *trivial complexity* represent the baseline models where no hyper-parameter was tuned. If the training has trails, then the best model from each trial is selected based on the one with the lowest validation loss. After a completed training the learning curves of training and validation losses are printed.

After the model selection, the one with the lowest validation loss is used to create an evaluation report with metrics and a confusion matrix. And finally, the ROC-AUC curve with the model predictions. The loss function used is the BCELoss() [Binary Cross Entropy] on the outputs of a Sigmoid() activation function to match the binary classification problem. All trials have been trained for 50 or 80 epochs with an Early Stopping mechanism of patience 5 and delta/tolerance of 0.0002 or in range(0.0001, 0.09, 0.0001).

2.1 LSTM Baseline model - GloVe embeddings

Baseline model architecture including the following layers in mostly default parameter values and using the LSTM cells:

(1) Embeddings layer

(1) LSTM layer (Bi-directional)

(1) Activation layer

(1) Dropout layer

(1) Linear output layer

Hyper-parameters in default values:

Batch_size (training, validation = 128)

Embedding_dim = 50

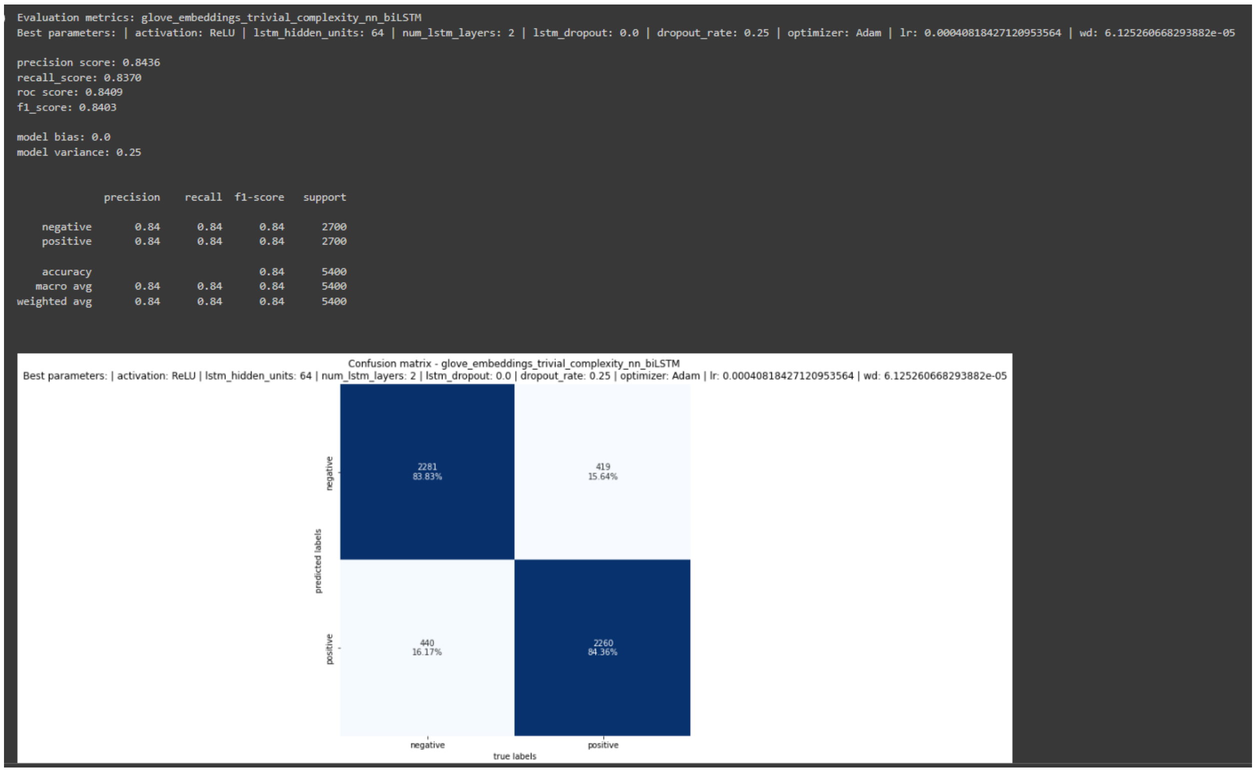
LSTM_hidden_units = 64

LSTM_layers = 2

Bi-directional = True

Dropout propability (lstm layer) = 0.15

Dropout propability (dense layer) = 0.25



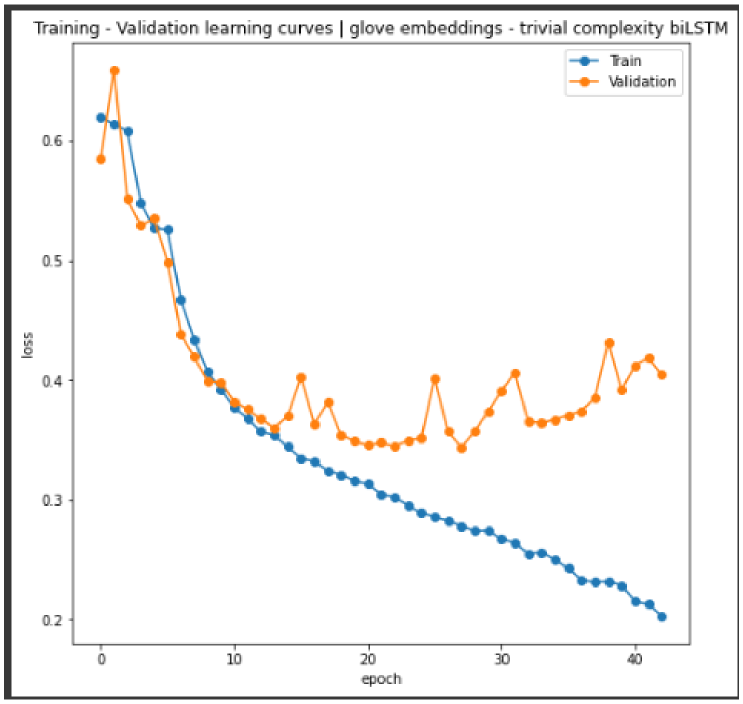
```
Evaluation metrics: glove_embeddings_trivial_complexity_nn_biLSTM
Best parameters: | activation: ReLU | lstm_hidden_units: 64 | num_lstm_layers: 2 | lstm_dropout: 0.0 | dropout_rate: 0.25 | optimizer: Adam | lr: 0.00040818427120953564 | wd: 6.125260668293882e-05

precision score: 0.8436
recall_score: 0.8370
roc score: 0.8409
f1_score: 0.8403

model bias: 0.0
model variance: 0.25


              precision    recall  f1-score   support

    negative       0.84      0.84      0.84      2700
    positive       0.84      0.84      0.84      2700

    accuracy                           0.84      5400
   macro avg       0.84      0.84      0.84      5400
weighted avg       0.84      0.84      0.84      5400
```

Confusion matrix - glove_embeddings_trivial_complexity_nn_biLSTM
Best parameters: | activation: ReLU | lstm_hidden_units: 64 | num_lstm_layers: 2 | lstm_dropout: 0.0 | dropout_rate: 0.25 | optimizer: Adam | lr: 0.00040818427120953564 | wd: 6.125260668293882e-05
predicted labels
negative
positive
2281 83.83%
419 15.64%
440 16.17%
2260 84.36%
true labels
negative
positive

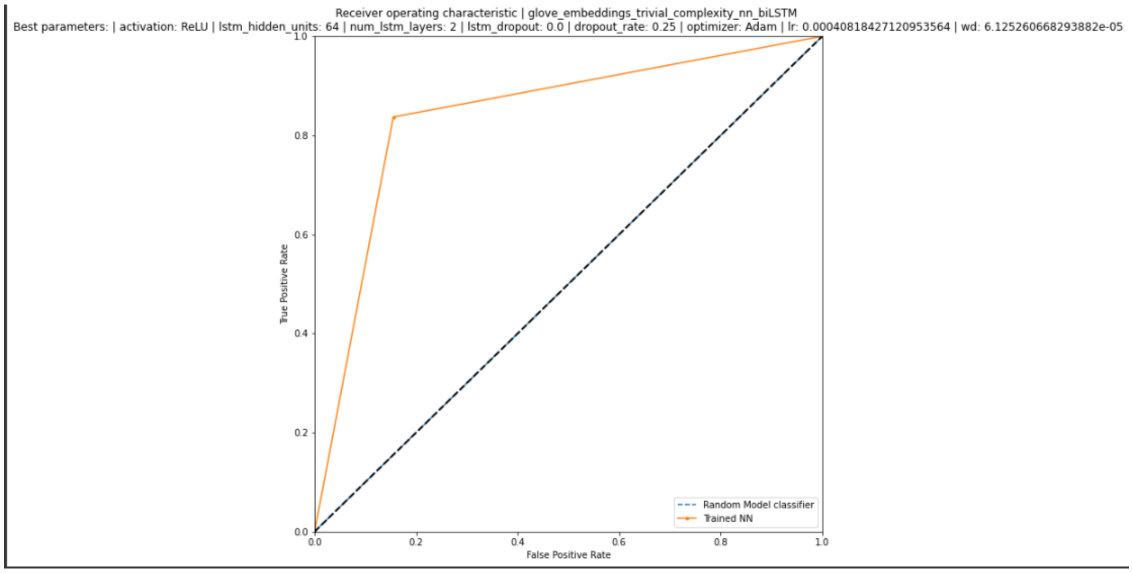
1-Evaluation Report | LSTM Baseline model - GloVe embeddings

FPR (False Positive Ratio): 16.17%

FNR (False Negative Ratio): 15.64%



2-Learning Curve | LSTM Baseline model - GloVe embeddings

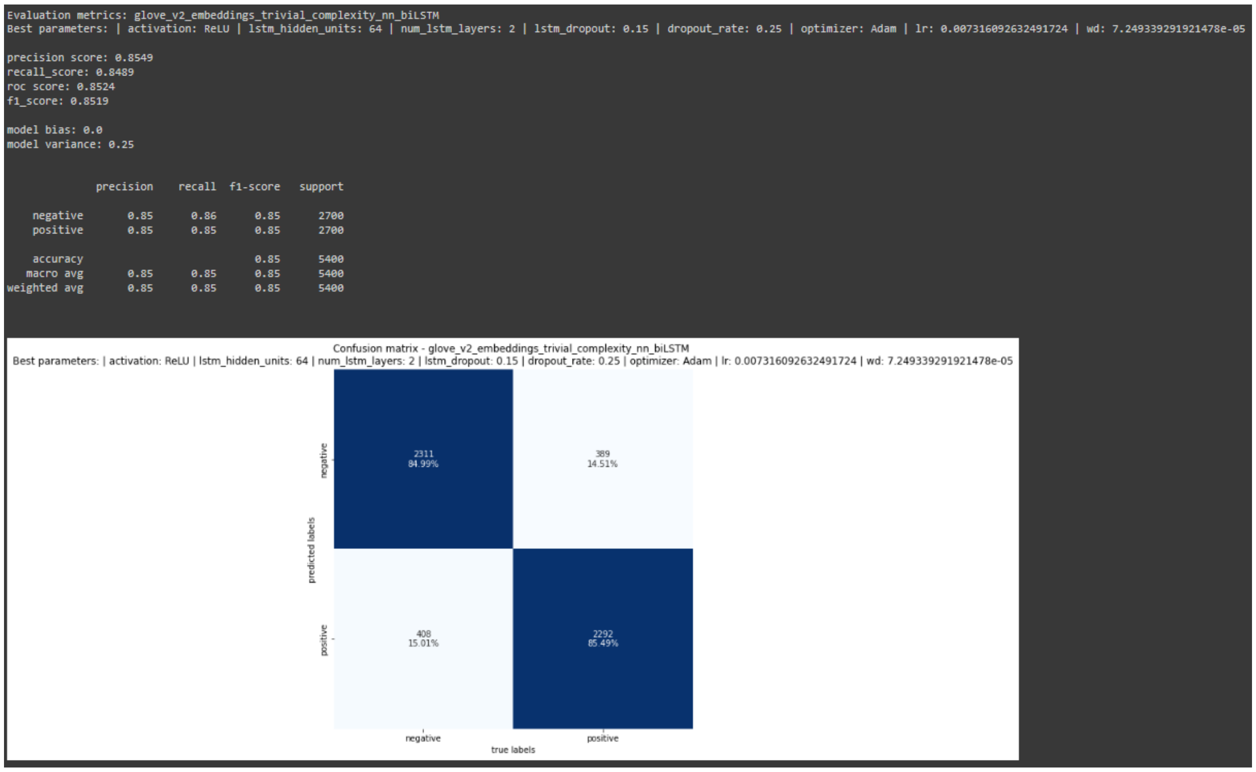


Best parameters: | activation: ReLU | lstm_hidden_units: 64 | num_lstm_layers: 2 | lstm_dropout: 0.0 | dropout_rate: 0.25 | optimizer: Adam | lr: 0.00040818427120953564 | wd: 6.125260668293882e-05

3-ROC Curve | LSTM Baseline model - GloVe embeddings

2.2 LSTM Baseline model - GloVe embeddings v2

The only difference with the previous model is the addition of a learning rate scheduler to control the changes of the learning rate every N epochs by a factor c. The number of epochs is set to 10 while the factor c is set to 0.5, which means halve of the learning every 10 epochs.



```
Evaluation metrics: glove_v2_embeddings_trivial_complexity_nn_biLSTM
Best parameters: | activation: ReLU | lstm_hidden_units: 64 | num_lstm_layers: 2 | lstm_dropout: 0.15 | dropout_rate: 0.25 | optimizer: Adam | lr: 0.007316092632491724 | wd: 7.249339291921478e-05

precision score: 0.8549
recall_score: 0.8489
roc score: 0.8524
f1_score: 0.8519

model bias: 0.0
model variance: 0.25

              precision    recall  f1-score   support

    negative       0.85      0.86      0.85      2700
    positive       0.85      0.85      0.85      2700

    accuracy                           0.85      5400
   macro avg       0.85      0.85      0.85      5400
weighted avg       0.85      0.85      0.85      5400
```

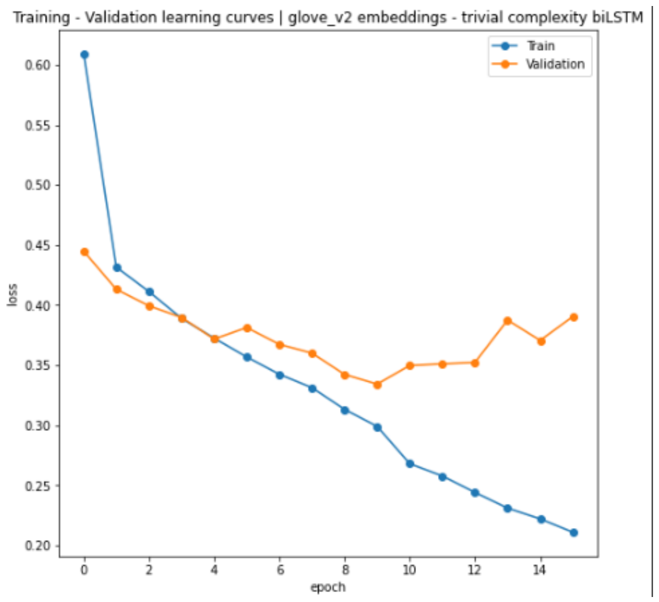
Confusion matrix - glove_v2_embeddings_trivial_complexity_nn_biLSTM
Best parameters: | activation: ReLU | lstm_hidden_units: 64 | num_lstm_layers: 2 | lstm_dropout: 0.15 | dropout_rate: 0.25 | optimizer: Adam | lr: 0.007316092632491724 | wd: 7.249339291921478e-05

4-Evaluation Report | LSTM Baseline model - GloVe embeddings v2

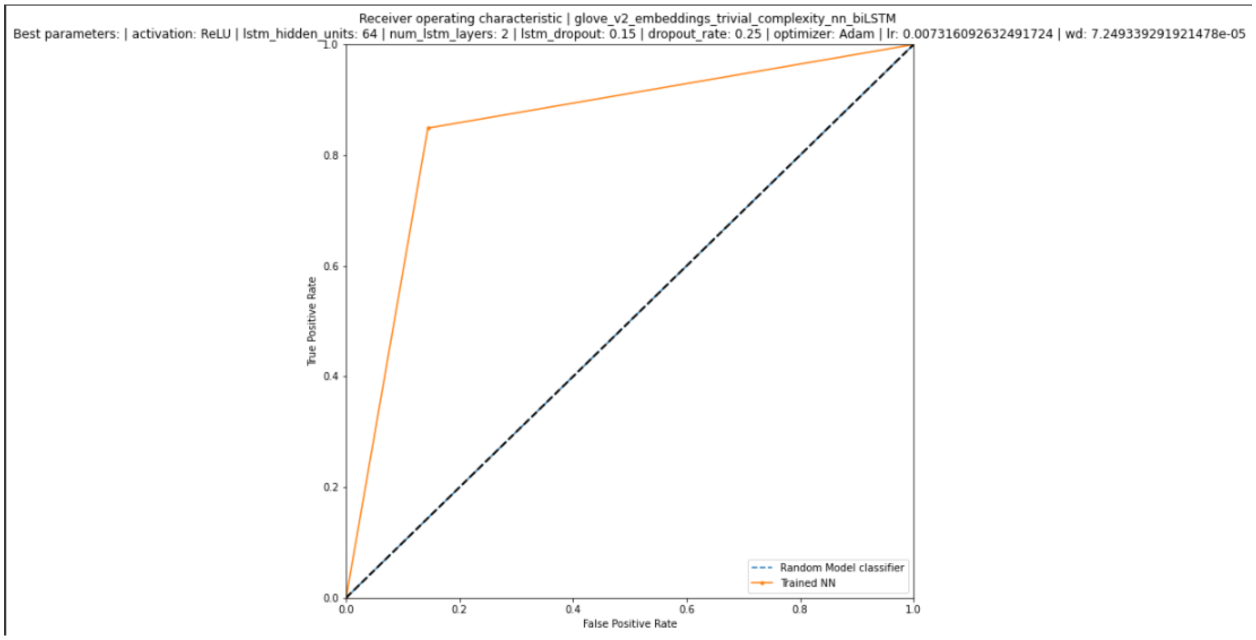
FPR (False Positive Ratio): 15.01% (improvement from experiment 2.1)

FNR (False Negative Ratio): 14.51% (improvement from experiment 2.1)

The learning rate scheduler helped the predictive performance of the neural network.



5-Learning Curve | LSTM Baseline model - GloVe embeddings v2

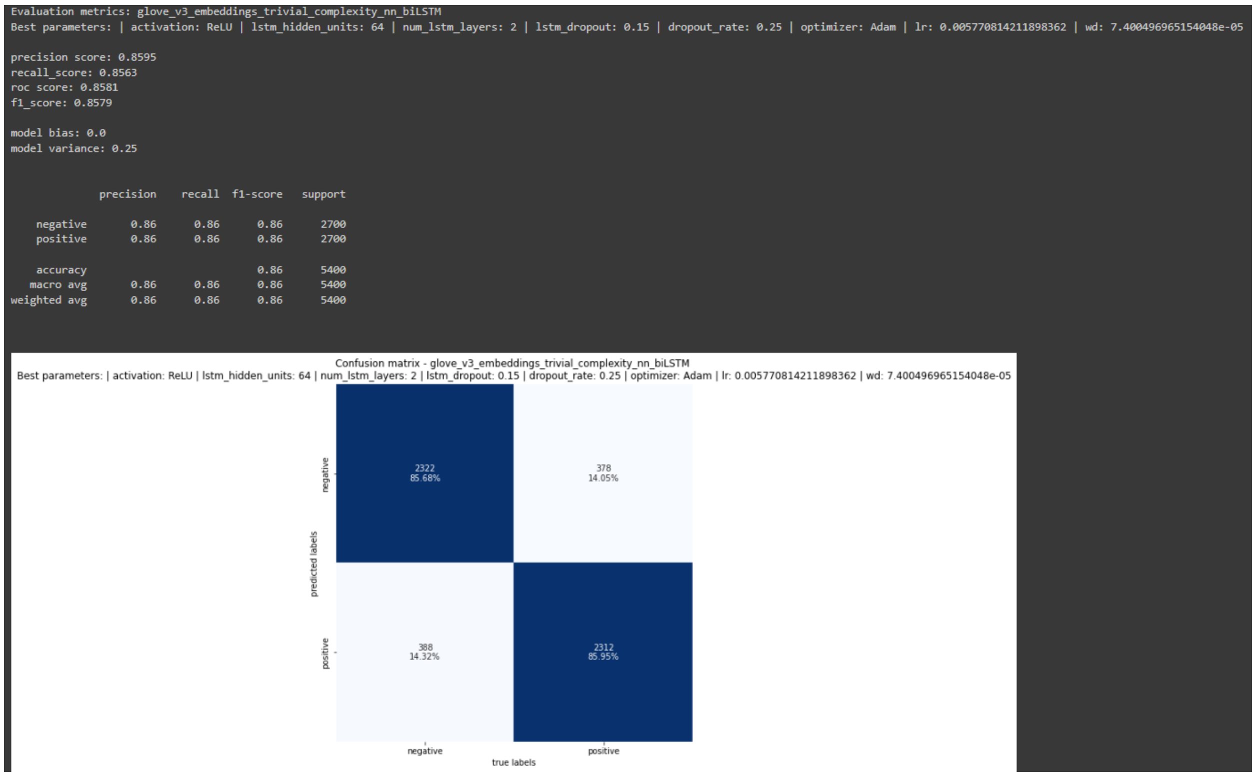


Best parameters: | activation: ReLU | lstm_hidden_units: 64 | num_lstm_layers: 2 | lstm_dropout: 0.15 | dropout_rate: 0.25 | optimizer: Adam | lr: 0.007316092632491724 | wd: 7.249339291921478e-05

6-ROC Curve | LSTM Baseline model - GloVe embeddings v2

2.3 LSTM Baseline model - GloVe embeddings v3

Based on the results of the previous two baseline GloVe models (2.1, 2.2) I have increased the number of epochs to adjust the learning rate from 10 to 20 epochs. And also, I have increased the gamma value of the learning rate scheduler from 0.5 to 0.85 in order to make smaller steps of changes in the learning rate. The rest of the hyper-parameters are the same like in the previous two experiments.



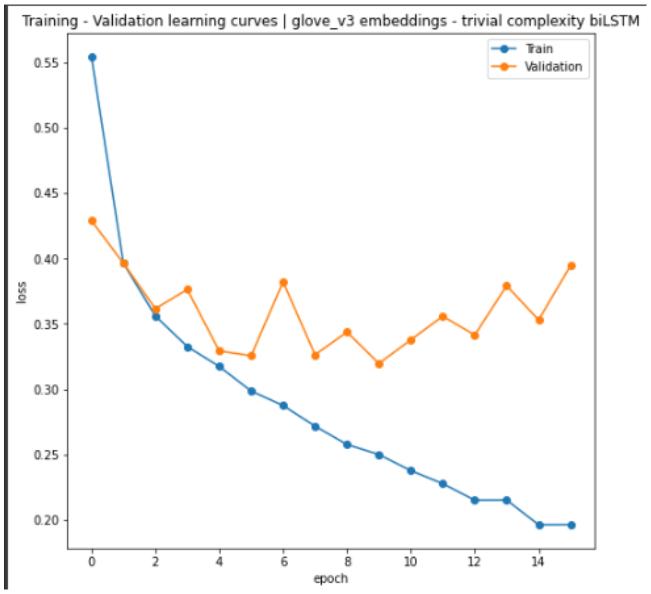
```
Evaluation metrics: glove_v3_embeddings_trivial_complexity_nn_biLSTM
Best parameters: | activation: ReLU | lstm_hidden_units: 64 | num_lstm_layers: 2 | lstm_dropout: 0.15 | dropout_rate: 0.25 | optimizer: Adam | lr: 0.005770814211898362 | wd: 7.400496965154048e-05

precision score: 0.8595
recall_score: 0.8563
roc score: 0.8581
f1_score: 0.8579

model bias: 0.0
model variance: 0.25


              precision    recall  f1-score   support

    negative       0.86      0.86      0.86      2700
    positive       0.86      0.86      0.86      2700

    accuracy                           0.86      5400
   macro avg       0.86      0.86      0.86      5400
weighted avg       0.86      0.86      0.86      5400
```

Confusion matrix - glove_v3_embeddings_trivial_complexity_nn_biLSTM
Best parameters: | activation: ReLU | lstm_hidden_units: 64 | num_lstm_layers: 2 | lstm_dropout: 0.15 | dropout_rate: 0.25 | optimizer: Adam | lr: 0.005770814211898362 | wd: 7.400496965154048e-05

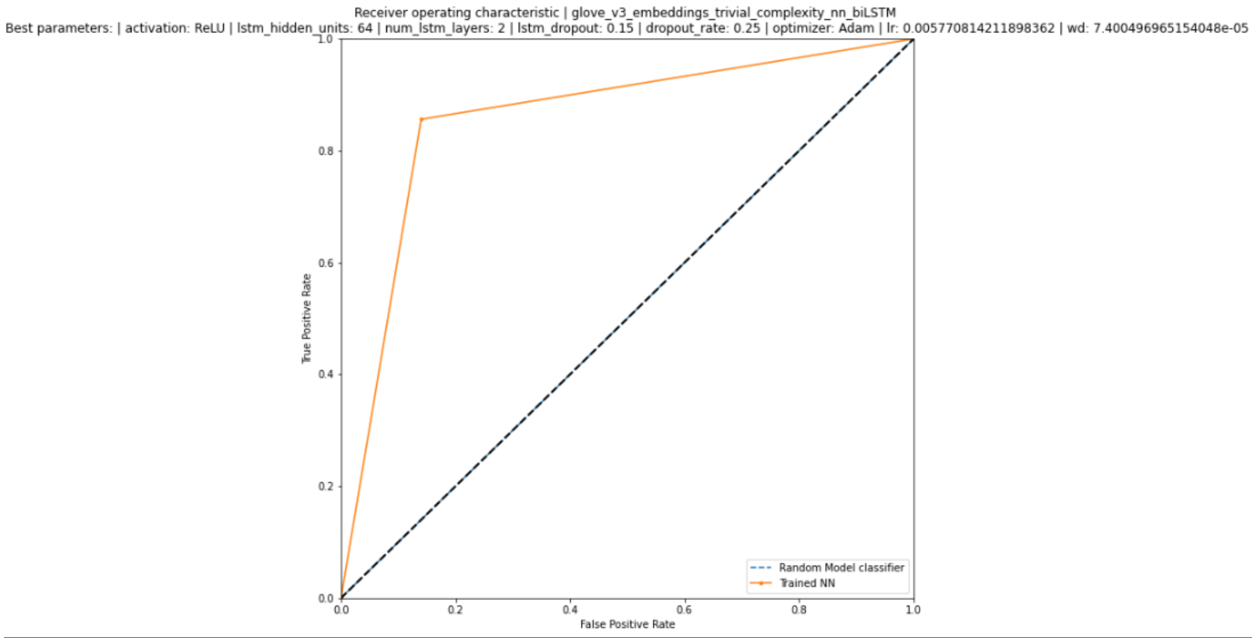
7-Evaluation Report | LSTM Baseline model - GloVe embeddings v3

FPR (False Positive Ratio): 14.32% (improvement from previous two experiments)

FNR (False Negative Ratio): 14.05% (improvement from previous two experiments)



8-Learning Curve | LSTM Baseline model - GloVe embeddings v3



Best parameters: | activation: ReLU | lstm_hidden_units: 64 | num_lstm_layers: 2 | lstm_dropout: 0.15 | dropout_rate: 0.25 | optimizer: Adam | lr: 0.005770814211898362 | wd: 7.400496965154048e-05

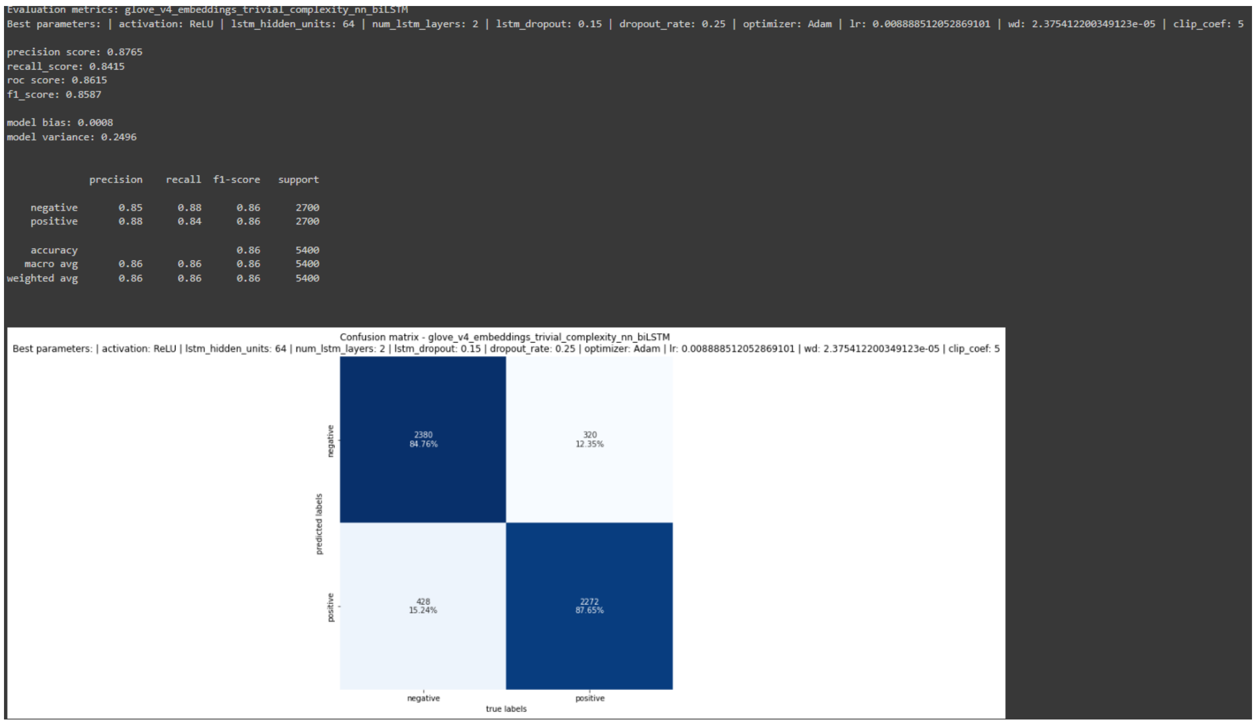
9-ROC Curve | LSTM Baseline model - GloVe embeddings v3

2.4 LSTM Baseline model - GloVe embeddings v4

Changes from the previous experiments:

- Add gradient clipping in the baseline model. Based on the prediction results, the training loss and the overall performance of the previous experiments it seems that gradient clipping it's not needed. However, for testing purposes I will apply it on the training of the fourth experiment.

- Dropout probability (LSTM layer) = 0.25 (from 0.15 - 3rd experiment).
- Dropout probability (dense layer) = 0.50 (from 0.25 - 3rd experiment).
- Drop the step size of the learning rate scheduler from 20 to 15 epochs. To change the learning rate sooner than in the third experiment.



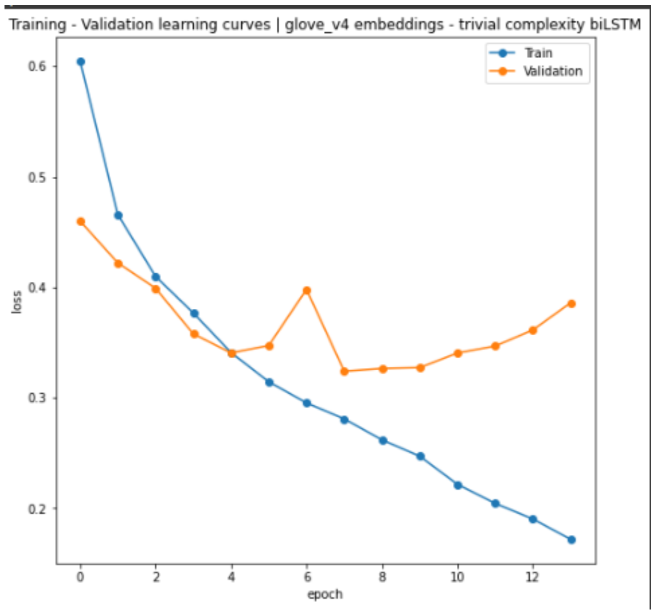
```
Evaluation metrics: glove_v4_embeddings_trivial_complexity_nn_biLSTM
Best parameters: | activation: ReLU | lstm_hidden_units: 64 | num_lstm_layers: 2 | lstm_dropout: 0.15 | dropout_rate: 0.25 | optimizer: Adam | lr: 0.008888512052869101 | wd: 2.375412200349123e-05 | clip_coef: 5

precision score: 0.8765
recall_score: 0.8415
roc score: 0.8615
f1_score: 0.8587

model bias: 0.0008
model variance: 0.2496


              precision    recall  f1-score   support

    negative       0.85      0.88      0.86      2700
    positive       0.88      0.84      0.86      2700

    accuracy                           0.86      5400
   macro avg       0.86      0.86      0.86      5400
weighted avg       0.86      0.86      0.86      5400
```

Confusion matrix - glove_v4_embeddings_trivial_complexity_nn_biLSTM
Best parameters: | activation: ReLU | lstm_hidden_units: 64 | num_lstm_layers: 2 | lstm_dropout: 0.15 | dropout_rate: 0.25 | optimizer: Adam | lr: 0.008888512052869101 | wd: 2.375412200349123e-05 | clip_coef: 5

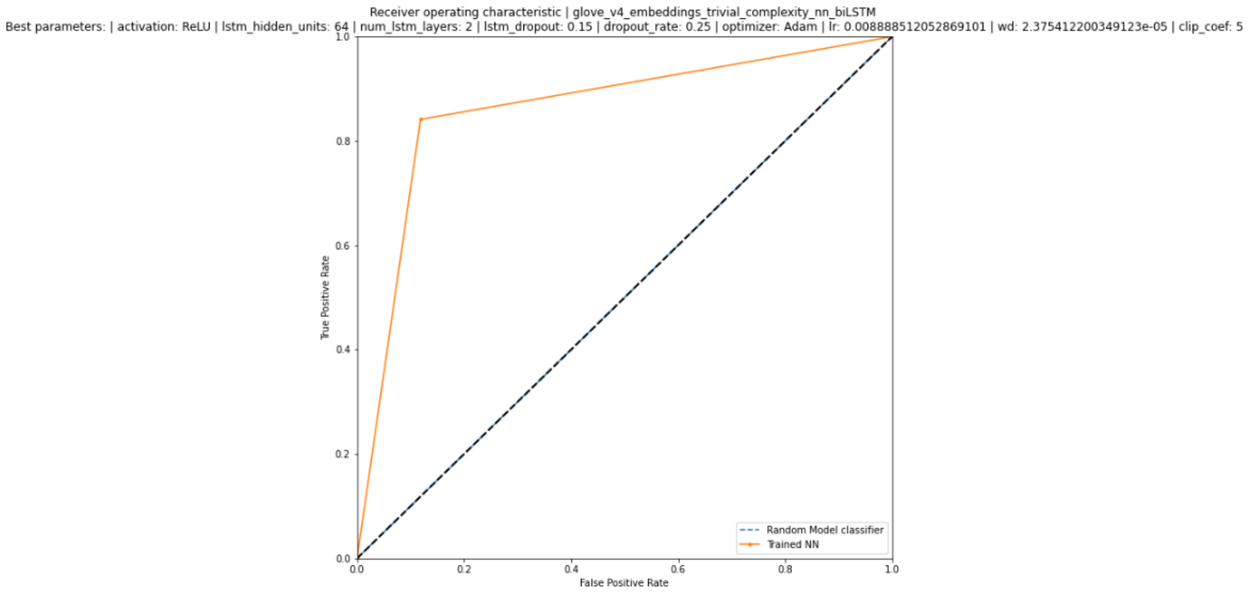
10- Evaluation Report | LSTM Baseline model - GloVe embeddings v4

FPR (False Positive Ratio): 15.24% (deterioration from previous two experiments)

FNR (False Negative Ratio): 12.35% (improvement from previous two experiments)



11-Learning Curve | LSTM Baseline model - GloVe embeddings v4



12-ROC Curve | LSTM Baseline model - GloVe embeddings v4

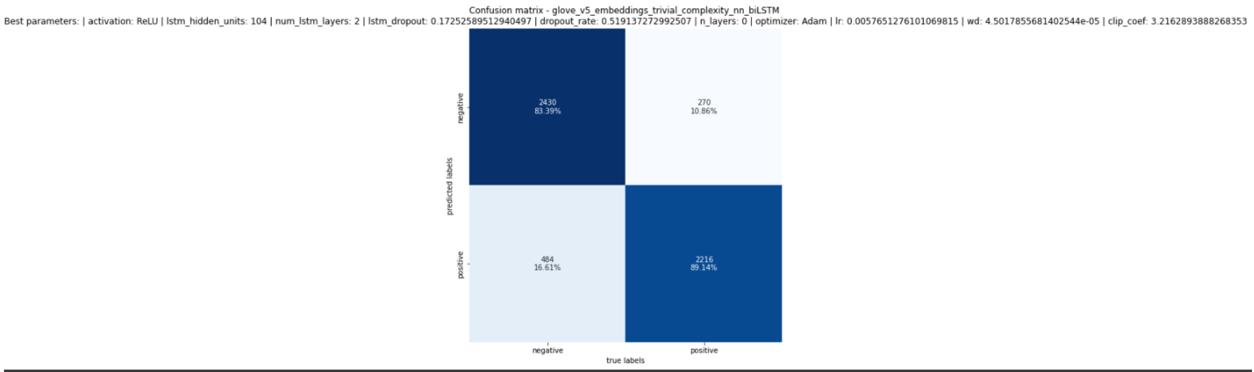
2.5 LSTM Advanced complexity model - GloVe embeddings v5

After the experimentation with the baseline model, I will proceed the hyper-parameter tuning of the baseline model architecture by also adding extra Linear layers as a hyper-parameter. For example, in the baseline model architecture I had only 1 nn.Linear layer. In the hyper-parameter tuning I will also test different scenarios with additional linear layers.

You may find below a brief description of the hyper-parameters tuned:

- lstm_hidden_units
- num_lstm_layers = number of stacked lstm layers
- lstm_dropout
- n_layers = number of hidden/dense nn.Linear layers
- lstm_hidden_units = hidden units of the linear layers
- dropout_probability of dropout layer
- activation function of the hidden layers
- learning rate
- weight decay if the learning rate
- clip coef of the gradient clipping method

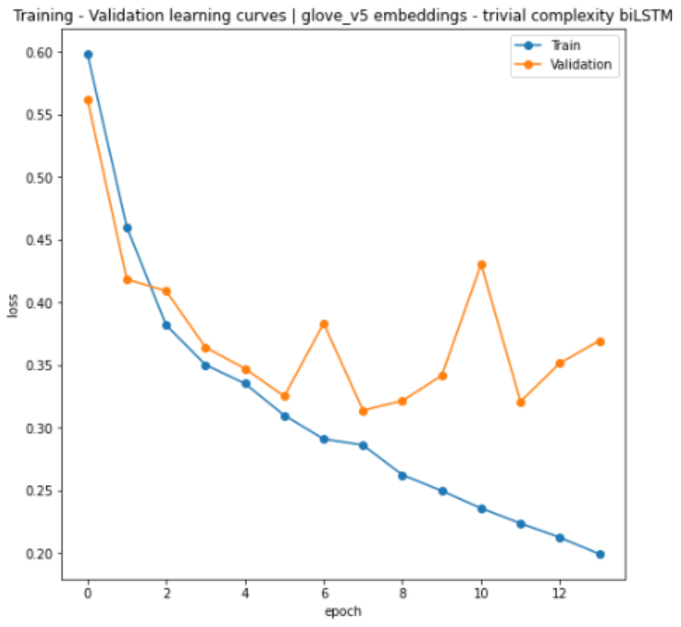
================================
Best model parameters:
================================
activation: ReLU
lstm_hidden_units: 104
num_lstm_layers: 2
lstm_dropout: 0.17252589512940497
dropout_rate: 0.519137272992507
n_layers: 0
optimizer: Adam
lr: 0.0057651276101069815
wd: 4.5017855681402544e-05
clip_coef: 3.2162893888268353


Confusion matrix - glove_v5_embeddings_trivial_complexity_nn_biLSTM

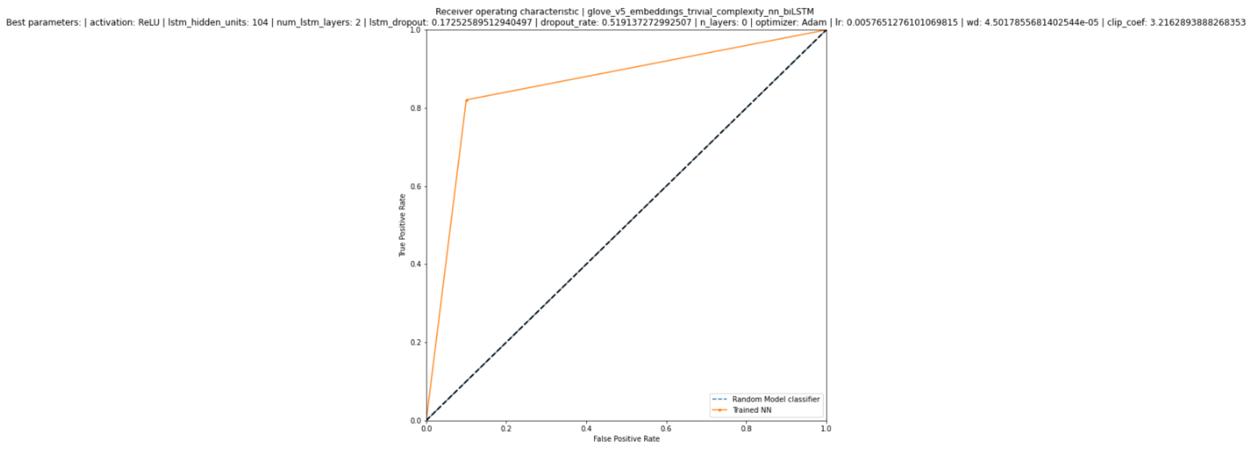
Best parameters: | activation: ReLU | lstm_hidden_units: 104 | num_lstm_layers: 2 | lstm_dropout: 0.17252589512940497 | dropout_rate: 0.519137272992507 | n_layers: 0 | optimizer: Adam | lr: 0.0057651276101069815 | wd: 4.5017855681402544e-05 | clip_coef: 3.2162893888268353

FPR (False Positive Ratio): 16.61% (deterioration from previous two experiments) -> Confuses negative reviews with positives.

FNR (False Negative Ratio): 10.86% (improvement from previous two experiments)



14-Learning Curve | LSTM Advanced complexity model - GloVe embeddings v5



Best parameters: | activation: ReLU | lstm_hidden_units: 104 | num_lstm_layers: 2 | lstm_dropout: 0.17252589512940497 | dropout_rate: 0.519137272992507 | n_layers: 0 | optimizer: Adam | lr: 0.0057651276101069815 | wd: 4.5017855681402544e-05 | clip_coef: 3.2162893888268353

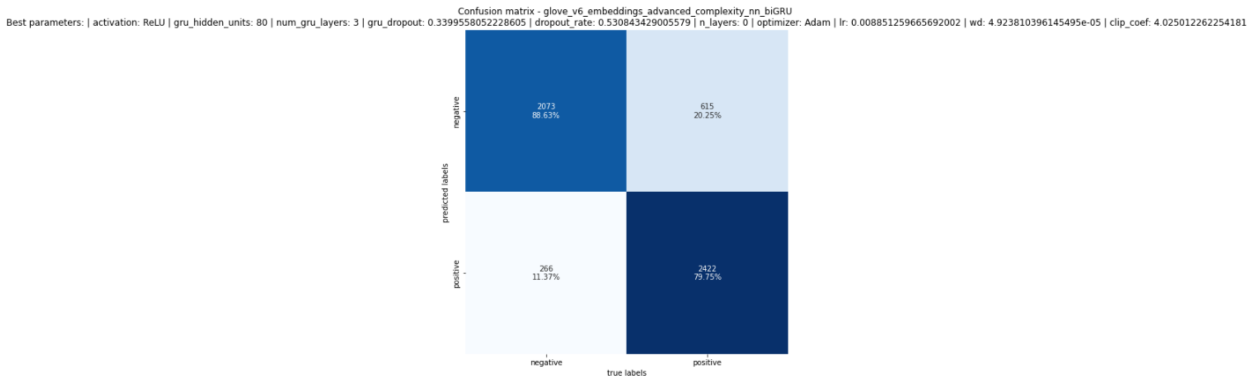
15-ROC Curve | LSTM Advanced complexity model - GloVe embeddings v5

2.6 GRU Advanced complexity model - GloVe embeddings v6

Same complexity as previous experiment, experiment 5. The main difference among this and the previous experiment is the use GRU cells instead of LSTM cells.

==============================

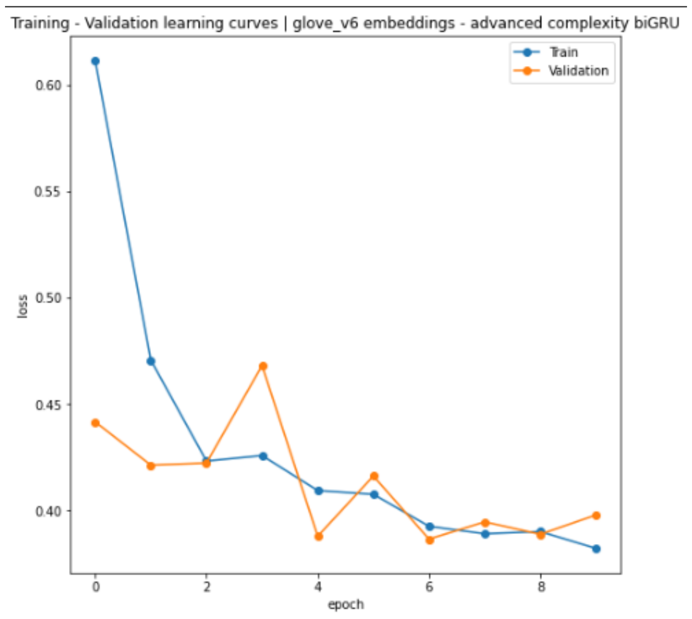
```
Best model parameters:
================================
activation: ReLU
gru_hidden_units: 80
num_gru_layers: 3
gru_dropout: 0.3399558052228605
dropout_rate: 0.530843429005579
n_layers: 0
optimizer: Adam
lr: 0.008851259665692002
wd: 4.923810396145495e-05
clip_coef: 4.025012262254181
```



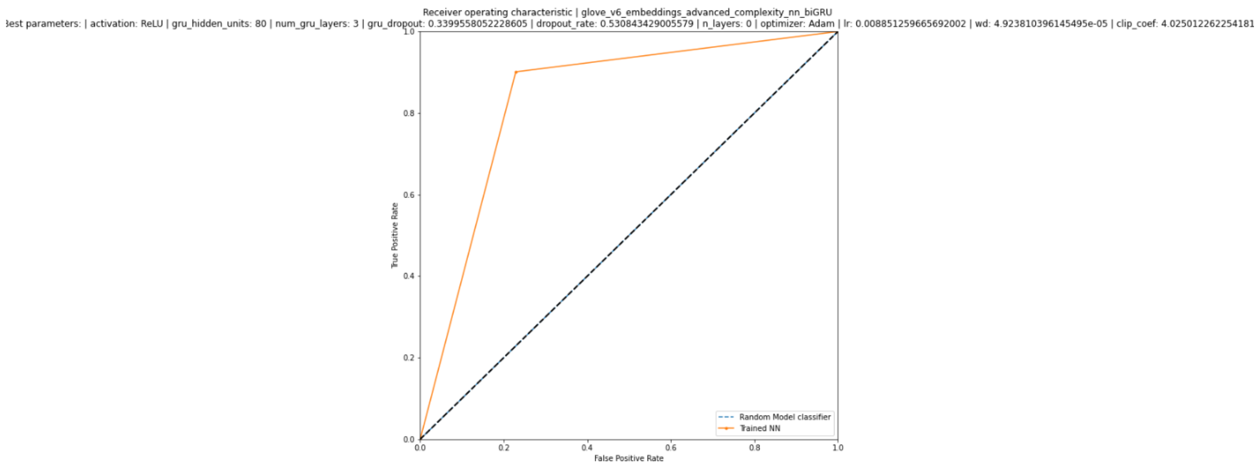
16- Evaluation Report | LSTM Advanced complexity model - GloVe embeddings v6

FPR (False Positive Ratio): 11.37% (improvement from any previous experiment)

FNR (False Negative Ratio): 20.25% (deterioration from all previous experiments) -> Confuses positive reviews with negatives. The result reveal that LSTM cells did better in identifying positive reviews while GRU did better in identifying negative reviews than any other model with LSTM cells.



17-Learning Curve | LSTM Advanced complexity model - GloVe embeddings v6



Best parameters: | activation: ReLU | gru_hidden_units: 80 | num_gru_layers: 3 | gru_dropout: 0.3399558052228605 | dropout_rate: 0.530843429005579 | n_layers: 0 | optimizer: Adam | lr: 0.008851259665692002 | wd: 4.923810396145495e-05 | clip_coef: 4.025012262254181

18-ROC Curve | LSTM Advanced complexity model - GloVe embeddings v6

3. Evaluation metrics table

Below I report the evaluation metric results from all the six experiments and I also comment on the results from all the previous figures to make a valid conclusion on the best RNN model.

Model name	Precision score	Recall score	Roc score	F1-score	Bias	Variance
LSTM Baseline model	0.8436	0.8370	0.8409	0.8403	0.0	0.25
LSTM Baseline model v2	0.8549	0.8489	0.8524	0.8519	0.0	0.25
LSTM Baseline model v3	0.8595	0.8563	0.8581	0.8579	0.0	0.25
LSTM Baseline model v4	0.8765	0.8415	0.8615	0.8587	0.0008	0.2496

LSTM Advanced complexity model v5	0.8914	0.8207	0.8604	0.8546	0.0032	0.2484
GRU Advanced complexity model v6	0.7975	0.9010	0.8361	0.8461	0.0084	0.2458

1-Evaluation metric results - RNN models

4. Conclusions

Based on the results of the above table and taking into consideration the reported evaluation metrics, learning curves and ROC curves I can conclude the following:

- The LSTM cells dominated the GRU cells in prediction performance.
- GRU cells models had a clear problem of exploding gradients, making necessary the application of gradient clipping. This conclusion is derived after looking the training per epoch of the model **Advanced complexity model - GloVe embeddings v6** in the Colab notebook.
- The fourth experiment had the best results with GRU models, of the sixth experiment, dominating in Recall scoring.
- Gradient clipping, dropout probabilities and learning rate scheduling helped the model training and predictive performance.
- All the models eventually overfitted at around 10 or 15 epochs. Which would raise a consideration for stricter regularization of the models.
- Even the high complexity models with many values of hyper-parameter tuning didn't improve with more than 1 linear layer. Both the fifth and sixth experiments had best models with number of extra linear layers = 0. Thus, one RNN layer and one Linear layer were sufficient for training good in predictive performance neural networks.
- From the fifth experiment I took a smaller sample of the 400,000 GloVe vectors to train the models. I noticed that by taking 400,000 vectors the 0.22% of the vocabulary was scored to the UNK token. While by taking 100,000 vectors (1/4 of the total GloVe vectors) the 0.44% of the vocabulary was only missing. Thus, instead of using 400,000 and increasing the number of iterations, I took a smaller sample of vectors. For more, please check the section **Import Vocabulary and Vectors from disk** from the Colab notebook.

5. Comparing the best model with the model(s) from Hw1 & Hw2

Model name	Precision score	Recall score	Roc score	F1-score	Bias	Variance
Hw3	0.8765	0.8415	0.8615	0.8587	0.0008	0.2496
Hw2	0.6589	0.9100	0.7194	0.7643	0.0726	0.2137
Hw1	0.8881	0.8880	-	0.8880	0.1379	0.2499

Table 2-Comparing models from the three homework exercises.

In terms of f1-score the baseline classifier of word2vec embeddings had almost 0.89 with the RRN neural network with LSTM cells coming very close. The logistic regression from the first homework dominates the rest of the Deep Learning models. However, the reported results of the first homework had a train-test split of 50-50%. In contrast to the 90-10% of the other two candidates. It would be interesting to test also the RRN model on a 50-50% train-test split. Last but not least, the model of the first homework had higher bias-variance values compared to the model from homework three. Thus, the third model was more accurate on the predictions when trained on 80% of samples instead of 50% of training samples.

6. Future work – Further improvements

1) Attention mechanism on both LSTM and GRU cells. Even though I had made some research on the attention mechanism and developed a few models. The training results weren't as expected; thus, no further work was reported. I have seen some implementations like those reported in the Reference section, however, the trained model had even worse predictive performance than the models trained.
2) Add more embedding dimensions for GloVe other than 50.

7. References

1) Implementing Attention Models in PyTorch | by Sumedh Pendurkar | Intel Student Ambassadors | Medium

2) NLP From Scratch: Translation with a Sequence to Sequence Network and Attention — PyTorch Tutorials 1.13.1+cu117 documentation

3) Pytorch-BiLSTM-attention-/model.py at main · korlankil/Pytorch-BiLSTM-attention- · GitHub

4) seq2seq/model.py at master · keon/seq2seq · GitHub

5) practical-pytorch/seq2seq-translation.ipynb at master · spro/practical-pytorch · GitHub

6) PyTorch-Batch-Attention-Seq2seq/attentionRNN.py at master · AuCson/PyTorch-Batch-Attention-Seq2seq · GitHub

7) Attention for sequence classification using a LSTM - nlp - PyTorch Forums

8) PyTorch - Bi-LSTM + Attention | Kaggle

9) ml-toolkit/rnn.py at master · chrisvdweth/ml-toolkit · GitHub

10) Sentiment analysis using LSTM - PyTorch | Kaggle

11) Sentiment Analysis with Pytorch — Part 4 — LSTM\BiLSTM Model | by Gal Hever | Medium

12) Sentiment Analysis with LSTM and TorchText with Code and Explanation (analyticsvidhya.com)