Name: MEDHA RUDRA

March 2022

1 Part 1: Sentiment Classification - Impact of RNN Architecture

1.1 Methods

The dataset used for this lab is the IMDB Large Movie Reviews dataset which has 50,000 samples. The dataset is already split into train set and test set in a 50/50 ratio, hence the experiments were conducted using the default split. For the purpose of this experiment, we loaded the data from tensorflow datasets and processed the reviews to convert it to a list of sequences. These sequences act as the input to our models. This dataset has two class labels: positive (represented as 1) and negative (represented as 0) and is ideal for binary sentiment classification problems.

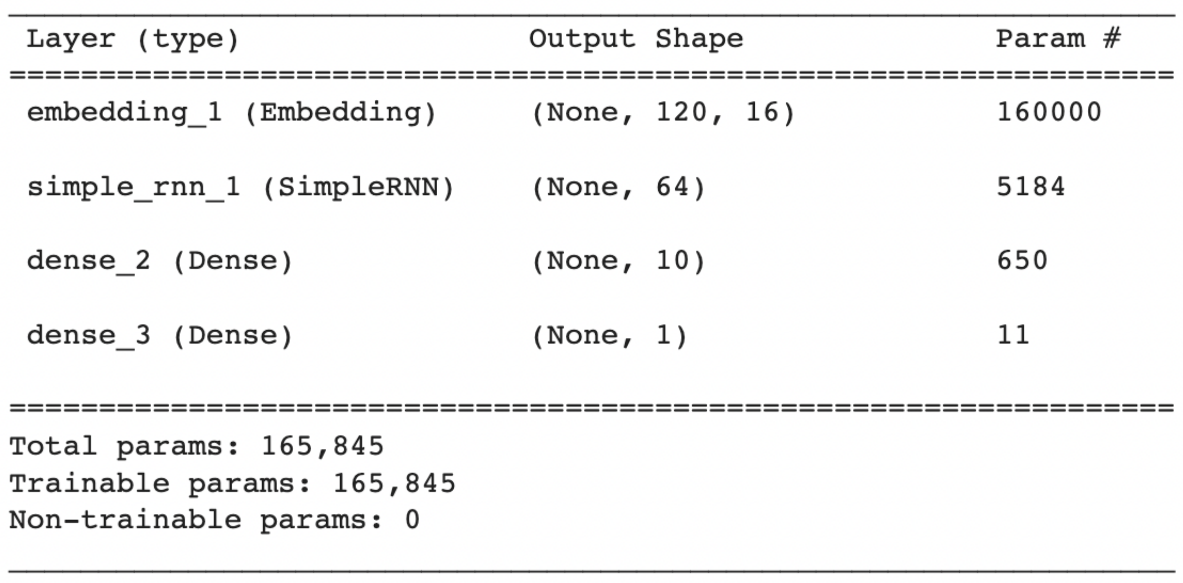
An example from the dataset is as follows:

Review: 'I have been known to fall asleep during films, but this is usually due to a combination of things including, really tired, being warm and comfortable on the sette and having just eaten a lot. However on this occasion I fell asleep because the film was rubbish. The plot development was constant. Constantly slow and boring. Things seemed to happen, but with no explanation of what was causing them or why. I admit, I may have missed part of the film, but i watched the majority of it and everything just seemed to happen of its own accord without any real concern for anything else. I cant recommend this film at all.'
Class Label: 0

This lab was conducted on Google Colab using a GPU on a system of 8GB RAM with an M1 chip. The model parameters used to train the models have been kept constant across all the models trained for this task. The vocabulary size had been set to 10000, embedding dimension was taken as 16. The maximum input length was set to 120 and the models were trained with a batch size of 64 over 10 epochs. Any unknown word was marked as 'UNK '.

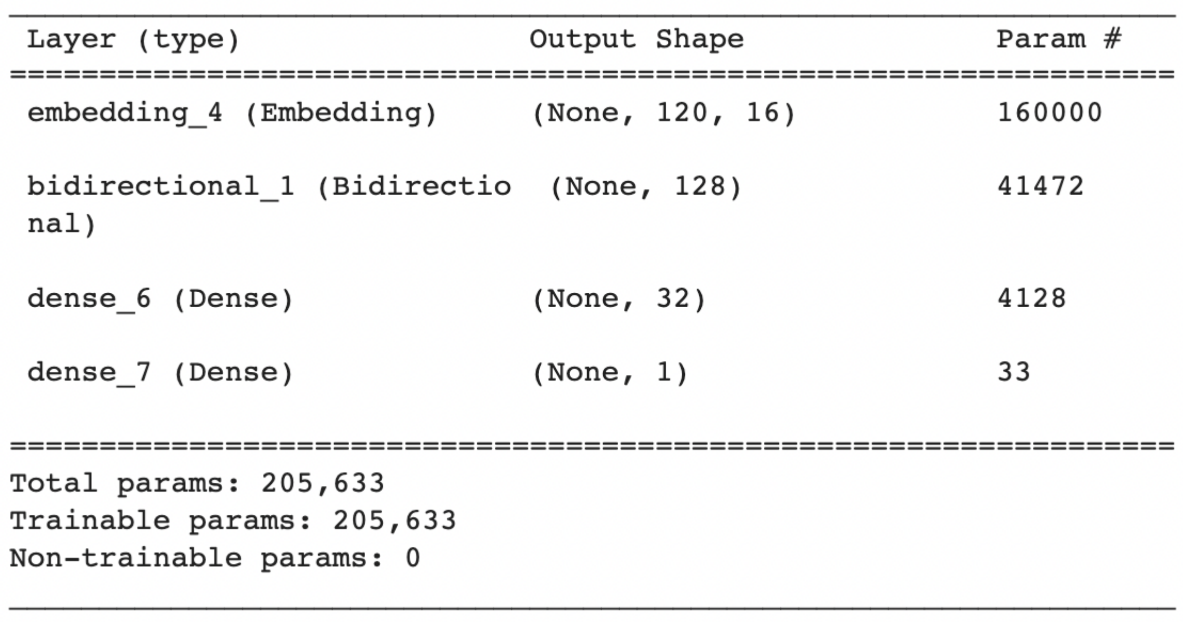
The loss function was computed using binary cross entropy and the optimizer applied to the models was the Adam optimizer with metric as 'accuracy'. The details of the three models trained are as follows:

Vanilla RNN Model:



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 120, 16)           160000

 simple_rnn_1 (SimpleRNN)    (None, 64)                5184

 dense_2 (Dense)             (None, 10)                650

 dense_3 (Dense)             (None, 1)                 11

=================================================================
Total params: 165,845
Trainable params: 165,845
Non-trainable params: 0
_________________________________________________________________
```

LSTM Model:



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_4 (Embedding)     (None, 120, 16)           160000

 bidirectional_1 (Bidirectio  (None, 128)              41472
 nal)

 dense_6 (Dense)             (None, 32)                4128

 dense_7 (Dense)             (None, 1)                 33

=================================================================
Total params: 205,633
Trainable params: 205,633
Non-trainable params: 0
_________________________________________________________________
```

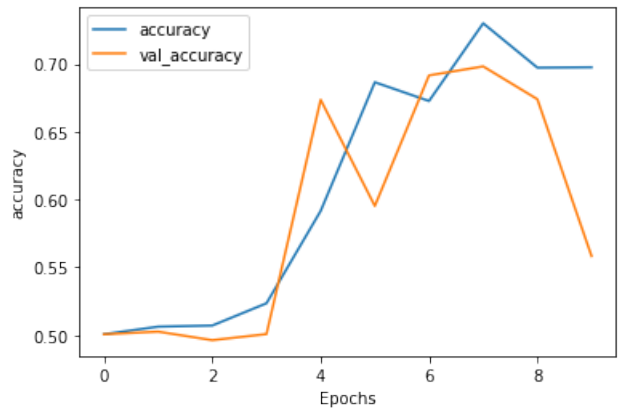
GRU Model:

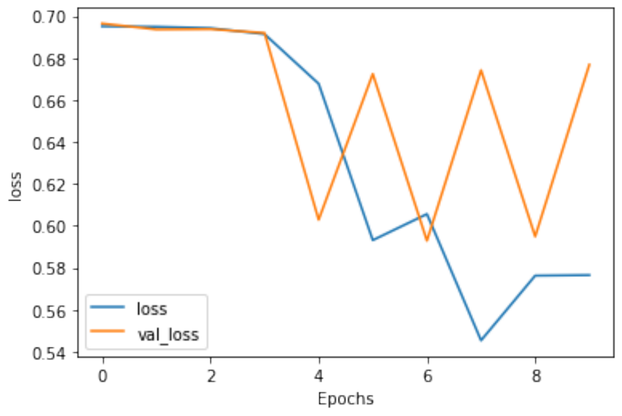
```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_3 (Embedding)     (None, 120, 16)           160000

 bidirectional (Bidirectiona  (None, 128)              31488
 l)

 dense_4 (Dense)             (None, 32)                4128

 dense_5 (Dense)             (None, 1)                 33

=================================================================
Total params: 195,649
Trainable params: 195,649
Non-trainable params: 0
_________________________________________________________________
```

1.2 Results

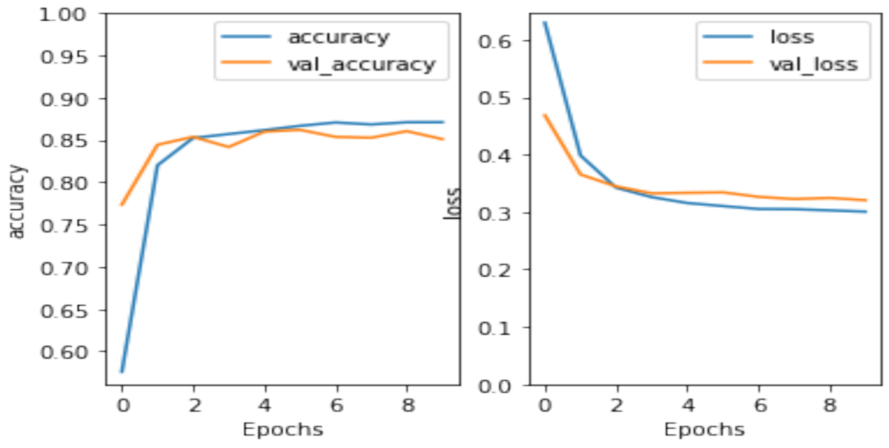
First, the models were trained and tested the usual way - training on 25,000 input data and testing on the remaining 25,000 data. The precision and recall obtained for each of the models in this case is:

Vanilla RNN:
Precision = 0.5451332920024798
Recall = 0.70344 The accuracy and loss of this model can be visualized as:

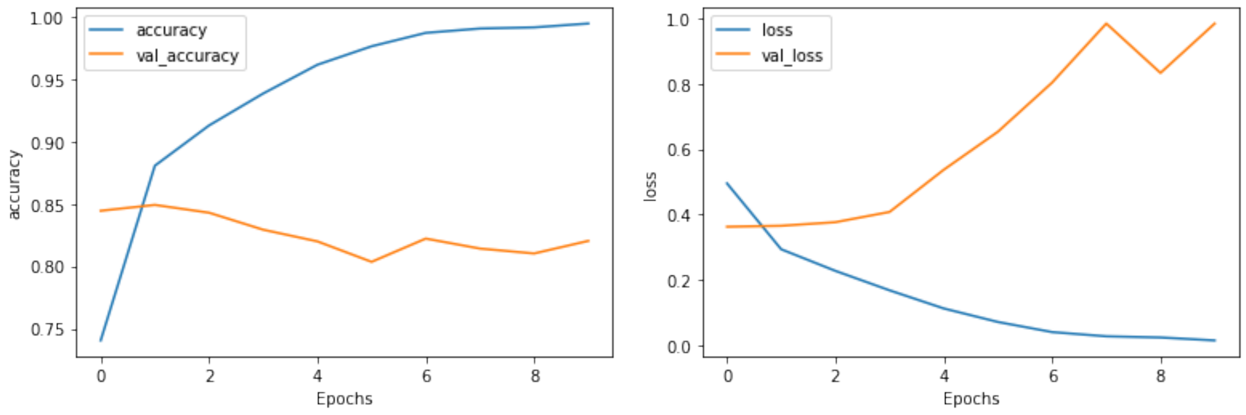




LSTM:
Precision = 0.7695146577536384
Recall = 0.88408 The accuracy and loss of this model can be visualized as:



GRU:
Precision = 0.826363414037662
Recall = 0.81096 The accuracy and loss of this model can be visualized as:



Then, we do a fine-grained analysis based on the input length and receive the following results. The set of small input consists of 8333 samples, medium input contains 8333 samples and the long inputs have 8334 samples. Here, small input refers to the input sequences of shortest length and long input refers to the input sequences of longest length.

Vanilla RNN Model:

=====SMALL INPUT======
Precision = 0.5822124071471592

```
Recall = 0.6708304418228083

=====MEDIUM INPUT=====
Precision = 0.5303951367781155
Recall = 0.6880236569738788

=====LONG INPUT=======
Precision = 0.5269328802039083
Recall = 0.7528526341344987
```

LSTM Model:

```
=====SMALL INPUT======
Precision = 0.8214360477287
Recall = 0.9077029840388618

=====MEDIUM INPUT======
Precision = 0.7567911714770797
Recall = 0.8787580088713652

=====LONG INPUT======
Precision = 0.7309113300492611
Recall = 0.864530225782957
```

GRU Model:

```
=====SMALL INPUT======
Precision = 0.868986454927604
Recall = 0.8607448531112654

=====MEDIUM INPUT======
Precision = 0.8142352347299344
Recall = 0.7949728930507639

=====LONG INPUT======
Precision = 0.7929405915983098
Recall = 0.7744598203447439
```

1.3 Analysis

From the experiments, it is observed that LSTMs and GRUs perform better than the Vanilla RNN both in terms of training time as well as precision and recall scores. For Vanilla RNNs, each epoch took a longer time to execute probably because it doesn't have a gated architecture. The gated RNNs would help in faster execution as the gates control the flow of information. In terms of the precision and recall scores, the gated RNNs performed better due to their ability to eliminate th eproblem of long delays. LSTMs and GRUs with their gated architecture solve the vanishing gradiend problem, avoiding the underflow and overflow errors.

If we observe the learning plots, we see that the LSTM model shows better learning. There's no overfitting or underfitting. The GRU seems to overfit on the validation data and the Vanilla RNN need more epochs to be able to learn the patterns of the data properly. GRUs may need dropout layers so that they don't over learn the patterns of the data and Vanilla RNNs would need gates for faster and accurate learning.

The performance trend observed when the models handle dshort, medium and long inputs are that the gated RNNs showed decrease in performance as the input sequence got longer. But the vanilla RNN performed better when the input sequence got longer. It seems that the presence of the the gated architechture controlled the information flow in such a way that the model could not perform that well when it saw a longer input sequence.

For mostly all the models the precision is slightly lower than the recall score. But for GRU, the precision and recall scores are similar. We can see that the GRU model overfit on this data. However, if either precision or recall score is significantly lower, like in case of vanilla RNN, the model is unable to learn at all. It is important to have both good precision and good recall scores so that model is able to predict unseen data correctly and also able to get a learning curve like that of the LSTM, without overfitting or underfitting.

2 Part 2: Sentiment Classification - Impact of Pre-trained Word Embedding

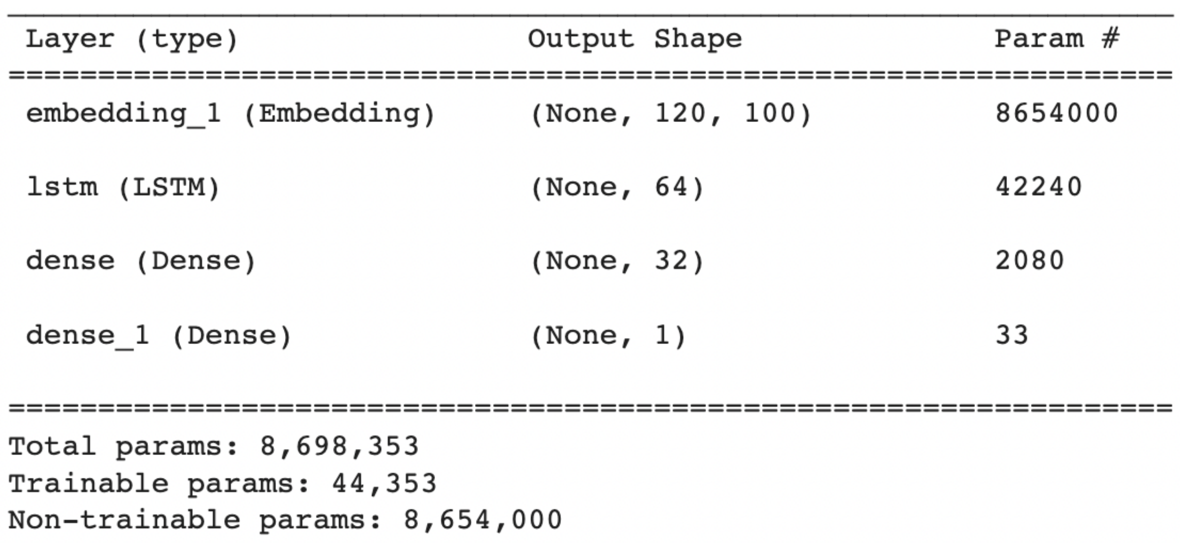
2.1 Methods

The same dataset has been used for this lab, that is, the IMDB Large Movie Reviews dataset which has 50,000 samples. The train/test split is in a 50/50 ratio by default, hence the experiments were conducted using the default split. For the purpose of this experiment, we loaded the data from tensorflow datasets and processed the reviews so that it can be fed to the pre-trained word embedding models like word2vec and GloVe. These models output a word embedding which is then fed to the embedding layer of our model. Finally we use

LSTM in our model to achieve the task of sentiment analysis.

This lab was conducted on Google Colab using a GPU on a system of 8GB RAM with an M1 chip. The model parameters used to train the models have been kept constant across all the models trained for this task. The vocabulary size had been set to the number of tokens in the input, embedding dimension was taken as 100. The maximum input length was set to 120 and the models were trained with a batch size of 64 over 20 epochs. Any unknown word was marked as 'UNK '. The loss function was computed using binary cross entropy and the optimizer applied to the models was the Adam optimizer with metric as 'accuracy'. The details of the two models trained are as follows:

LSTM Pre-trained on Word2Vec:



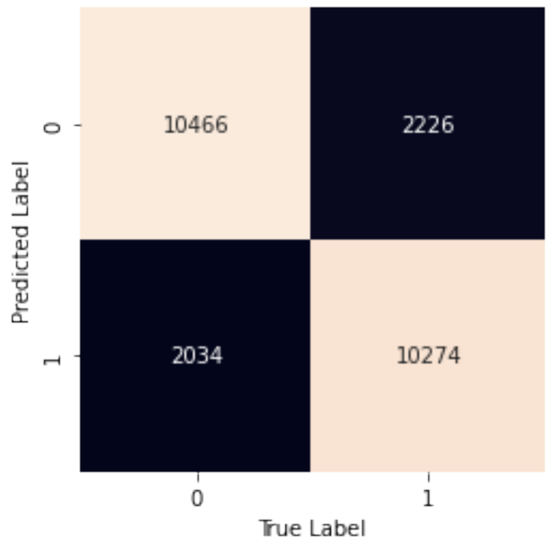
```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 120, 100)          8654000

 lstm (LSTM)                 (None, 64)                42240

 dense (Dense)               (None, 32)                2080

 dense_1 (Dense)             (None, 1)                 33

=================================================================
Total params: 8,698,353
Trainable params: 44,353
Non-trainable params: 8,654,000
```

LSTM Pre-trained on GloVe:

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_2 (Embedding)     (None, 120, 100)          8654000

 lstm_2 (LSTM)               (None, 64)                42240

 dense_4 (Dense)             (None, 32)                2080

 dense_5 (Dense)             (None, 1)                 33

=================================================================
Total params: 8,698,353
Trainable params: 44,353
Non-trainable params: 8,654,000
_________________________________________________________________
```

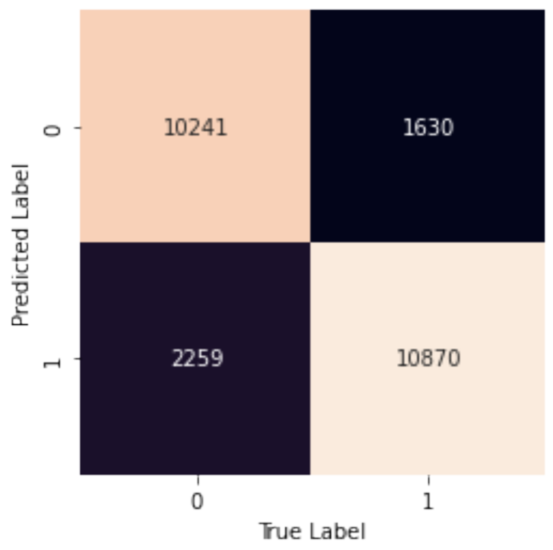
2.2 Results

The first model using LSTM is pre-trained using Word2Vec and the results received are as follows:
Precision = 0.8347416314592135
Recall = 0.82192
Confusion Matrix:



The second model using LSTM is pre-trained using GloVe and the results received are as follows:

Precision = 0.8279381521821921
Recall = 0.8696
Confusion Matrix:



2.3 Analysis

The precision score for both the models was approximate around 83%, which means whenever each of these models predicts if a sentiment is positive, it is correct 83% of the times. So, they could equally competent in predicting the proportion of positive identifications that was actually correct which is evident from the confusion matrix where the number of true positives and false positives for both models is similar (approximately 10,000 and 2000 respectively). The recall score of the model pre-trained on GloVe, however, produced a slightly better result of 87% as compared to the model pre-trained on Word2Vec which had a recall of 82%. Again, this is evident from the confusion matrix where the difference in both models is in terms of false negatives for the recall score. The model pre-trained with GloVe had lesser false negatives.

LSTM Pre-trained on Word2Vec: In this model, the precision and recall have similar score, that is, around 83%. This seems to be a good model because the model has the ability to find relevant data samples and is capable of classifying relevant data correctly upto 83%.
LSTM Pre-trained on GloVe: The recall of this model is slightly better than it's precision. The model is hence able to find the relevant data samples, but it is not as good in classifying the relevant data.If we try to increase the precision by modifying the classification threshold (in this case, 0.5), then recall would decrease. So we can marginally modify the threshold so that the precision and recall of the model is similar in score and at the same time equally good in terms of score.

Word2Vec creates embeddings that are able to capture if two words appear in the same context. Whereas GloVe objectively tries to predict the probability of two words occuring together. The IMDB dataset probably has the data structured in a way that the reviews have certain words appearing together with helps the model identify easily if the occur together and predict the class (positive or negative sentiment) accordingly. Overall, we need both a good precision score and a good recall score. If precision score is good but recall isn't, then false positives would reduce but false negatives would be high and vice versa. It's preferable to have a model with a balance between the two. The confusion metric helps us get an insight of both precision and recall and seems to provide more details in tabular format and hence seems to be more helpful than just having the precision score or just the recall score.

The performance of these models is better as compared to the performance of the baseline models that use the tokenized representations as input instead. In terms of the training time, pre-trained models took significantly less time than the baseline models because less parameters had to be learnt. Also, in terms of the precision and recall scores, the pre-trained models did perform slightly better may be because the token representations in our models were unable to grasp the semantic patterns required for the IMDB dataset. We should also consider the fact that the baseline models were trained up to 10 epochs whereas the models with pre-trained embeddings where trained up to 20 epchs. Increasing the number of epochs for the baseline models may help in levelling up it's performance and learn the dataset better over more time.