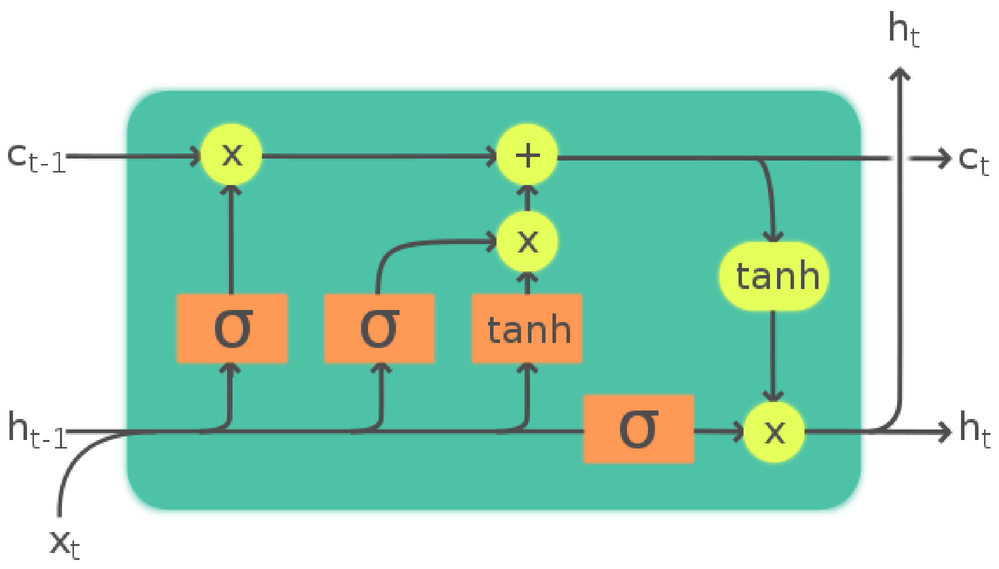
IMDB Sentiment Analysis | LSTM



Shubham Kaushal
Contact: +91 7045 603 792
shubhamkaushal765@gmail.com
Gmail | LinkedIn | GitHub

Table of contents

Dataset

IMDB is a dataset of 50,000 movies reviews labeled by sentiment (positive/negative). Instances have been preprocessed, and each review is encoded as a list of word indexes (integers). Words are indexed based on their frequency on the dataset. The dataset is split into a training set (25,000) and a testing set (25,000).

The task is to perform sentiment analysis to classify the instances into positive or negative reviews using Recurrent Neural Networks (RNNs).

Exploring the dataset

Let us look at some of the reviews:
https://builtin.com/data-science/how-build-neural-network-keras

Preprocessing

The input reviews must be converted to integers before feeding them into a model. For this preprocessing step, word embedding or vectorization is used. Two ways are explored in this project. The first is vectorizing using a matrix. Each row represents each instance and each column represents a unique word. This technique takes a huge amount of memory but is faster and easier to train.

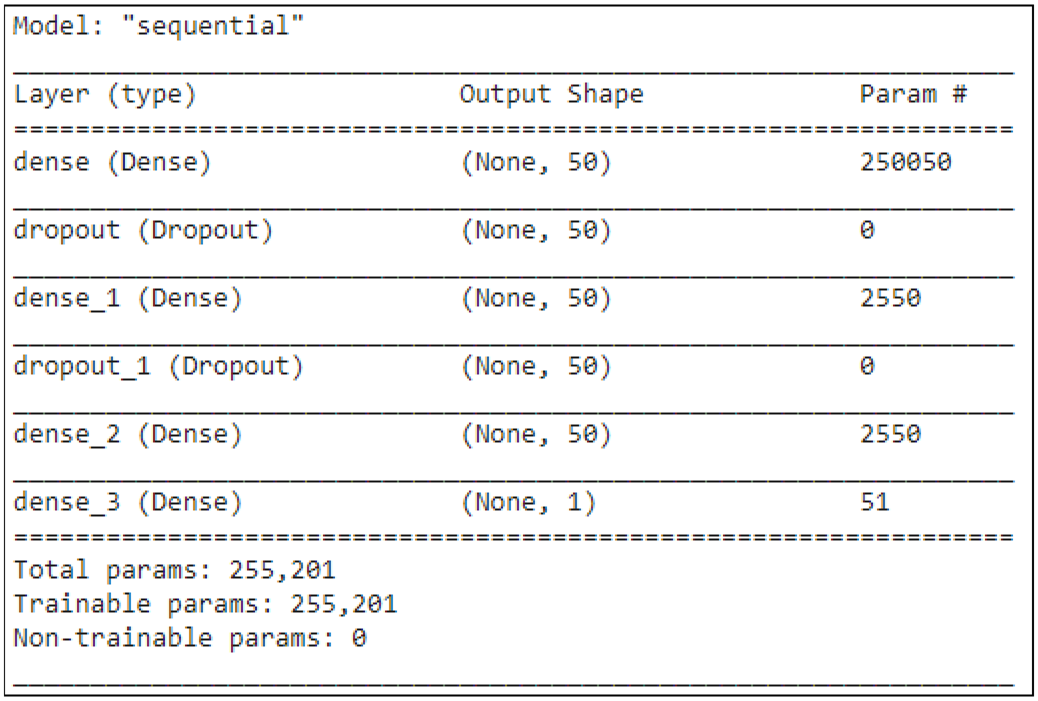
The second is using the Embedding layer and LSTM. Keras provides for both layers to be included in the model. It takes less memory, but the model is more complicated and training time is larger.

For the second technique, another thing that must be taken care of is that reviews are of different lengths. Before feeding them to a model, they must be cast to the same size. For this, zero padding is used. In this technique, a maximum length is defined. Reviews with a length greater than this are truncated, and those with a smaller length get zero-padded.

Matrix representation & Dense model

The data is represented (vectorized) as a matrix. The number of rows is equal to the number of instances, and the number of columns is equal to the total number of unique words. It is apparent that with a large dataset and with many unique words, the matrix becomes huge. Hence, in this project, only 5,000 unique words are used with 50,000 instances. Data is split into 40,000 to 10,000 (training to testing). We also see that this model’s training time is minimal and provides good accuracy.

An MLP is used to train on the matrix. The model summary is as follows:



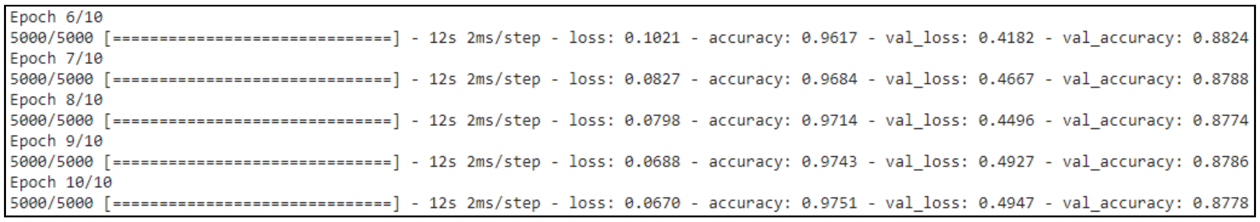
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 50)                250050
_________________________________________________________________
dropout (Dropout)            (None, 50)                0
_________________________________________________________________
dense_1 (Dense)              (None, 50)                2550
_________________________________________________________________
dropout_1 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_2 (Dense)              (None, 50)                2550
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 51
=================================================================
Total params: 255,201
Trainable params: 255,201
Non-trainable params: 0
_________________________________________________________________
```

Results from MLP

Different optimizers (Adam, SGD, RMSprop) were used to compile the model and select the model with the best performance. The performances of the optimizers are:

Adam Optimizer

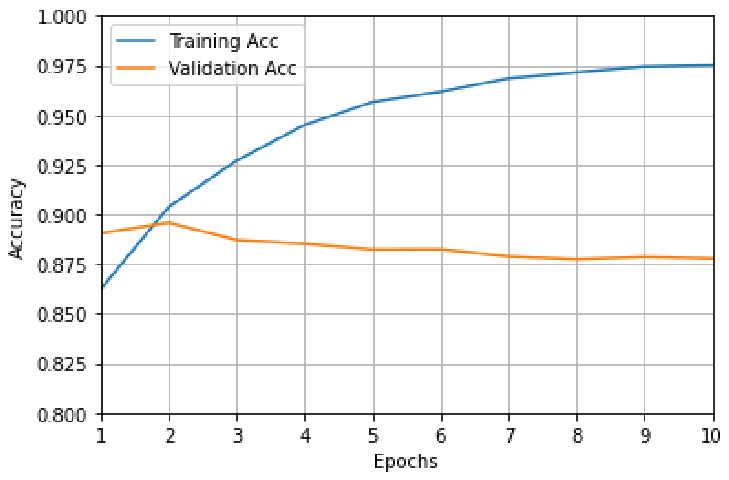
The losses and accuracy on the training set and testing set are:



```
Epoch 6/10
5000/5000 [==============================] - 12s 2ms/step - loss: 0.1021 - accuracy: 0.9617 - val_loss: 0.4182 - val_accuracy: 0.8824
Epoch 7/10
5000/5000 [==============================] - 12s 2ms/step - loss: 0.0827 - accuracy: 0.9684 - val_loss: 0.4667 - val_accuracy: 0.8788
Epoch 8/10
5000/5000 [==============================] - 12s 2ms/step - loss: 0.0798 - accuracy: 0.9714 - val_loss: 0.4496 - val_accuracy: 0.8774
Epoch 9/10
5000/5000 [==============================] - 12s 2ms/step - loss: 0.0688 - accuracy: 0.9743 - val_loss: 0.4927 - val_accuracy: 0.8786
Epoch 10/10
5000/5000 [==============================] - 12s 2ms/step - loss: 0.0670 - accuracy: 0.9751 - val_loss: 0.4947 - val_accuracy: 0.8778
```

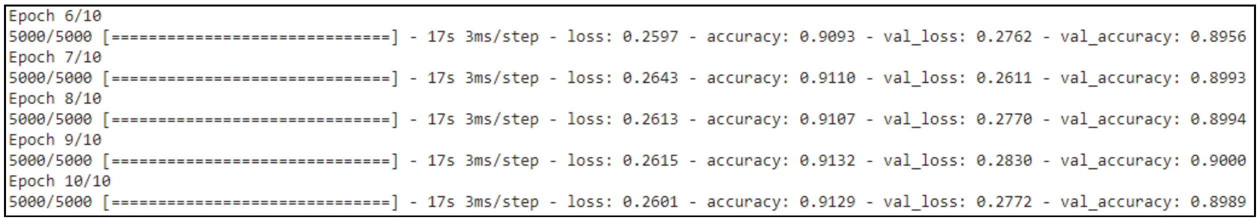
The adam optimizer performs quite well with over 87% accuracy in the test data. The accuracy for the training set is above 97% (below graph). The separation is quite large. This is due to overfitting. Hence, the model needs to be regularized.

Accuracy on Training and Testing data:



RMSprop Optimizer

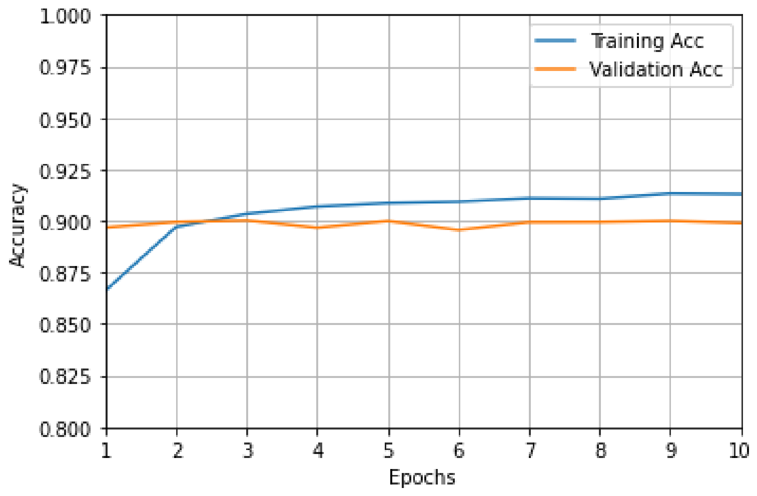
The losses and accuracy on the training set and testing set are:



```
Epoch 6/10
5000/5000 [==============================] - 17s 3ms/step - loss: 0.2597 - accuracy: 0.9093 - val_loss: 0.2762 - val_accuracy: 0.8956
Epoch 7/10
5000/5000 [==============================] - 17s 3ms/step - loss: 0.2643 - accuracy: 0.9110 - val_loss: 0.2611 - val_accuracy: 0.8993
Epoch 8/10
5000/5000 [==============================] - 17s 3ms/step - loss: 0.2613 - accuracy: 0.9107 - val_loss: 0.2770 - val_accuracy: 0.8994
Epoch 9/10
5000/5000 [==============================] - 17s 3ms/step - loss: 0.2615 - accuracy: 0.9132 - val_loss: 0.2830 - val_accuracy: 0.9000
Epoch 10/10
5000/5000 [==============================] - 17s 3ms/step - loss: 0.2601 - accuracy: 0.9129 - val_loss: 0.2772 - val_accuracy: 0.8989
```

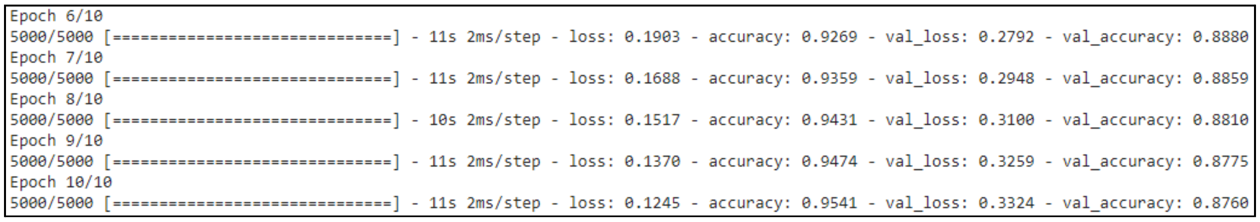
The RMSprop optimizer works better than the adam optimizer. The accuracy of the testing set is ~ 90% and that of the training set is ~ 91%. The separation in the below graph is very small. Hence this model performs better than the previous one without severely overfitting the data.

Accuracy on Training and Testing data:



SGD Optimizer

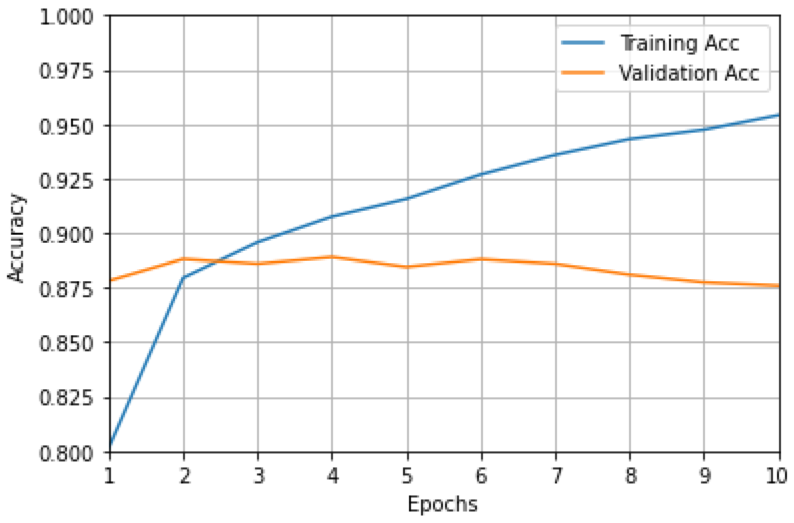
The losses and accuracy on the training set and testing set are:



```
Epoch 6/10
5000/5000 [==============================] - 11s 2ms/step - loss: 0.1903 - accuracy: 0.9269 - val_loss: 0.2792 - val_accuracy: 0.8880
Epoch 7/10
5000/5000 [==============================] - 11s 2ms/step - loss: 0.1688 - accuracy: 0.9359 - val_loss: 0.2948 - val_accuracy: 0.8859
Epoch 8/10
5000/5000 [==============================] - 10s 2ms/step - loss: 0.1517 - accuracy: 0.9431 - val_loss: 0.3100 - val_accuracy: 0.8810
Epoch 9/10
5000/5000 [==============================] - 11s 2ms/step - loss: 0.1370 - accuracy: 0.9474 - val_loss: 0.3259 - val_accuracy: 0.8775
Epoch 10/10
5000/5000 [==============================] - 11s 2ms/step - loss: 0.1245 - accuracy: 0.9541 - val_loss: 0.3324 - val_accuracy: 0.8760
```

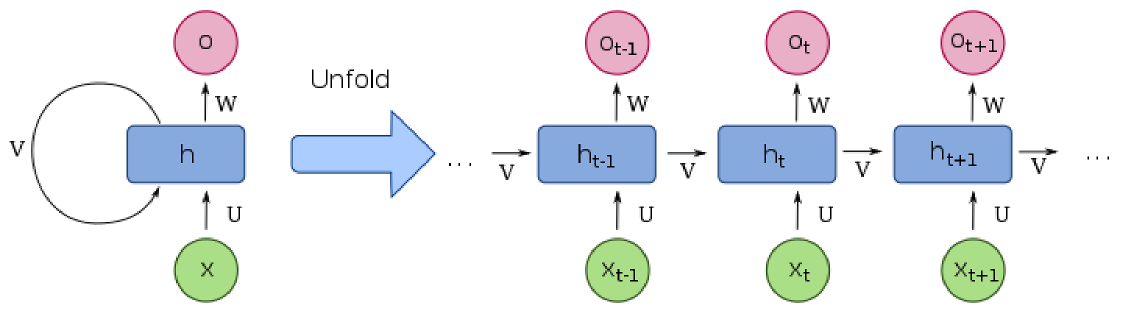
The SGD optimizer also performs well with over 87% accuracy in the test data. The accuracy for the training set is above 95% (below graph). The separation grows large. This is due to overfitting. Hence, the model needs to be regularized.

Accuracy on Training and Testing data:



Recurrent Neural Networks

A recurrent neural network is a type of artificial neural network commonly used in speech recognition and natural language processing.

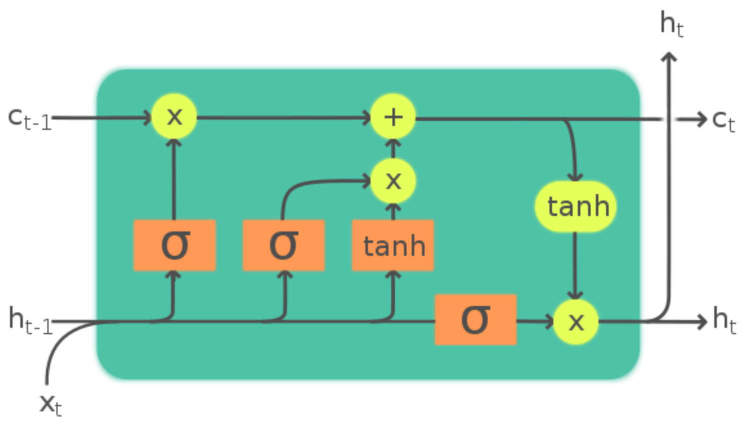


Basic Recurrent Neural Network, Source: Wikipedia

It forms a directed graph along a temporal sequence, allowing it to exhibit temporal dynamic behavior. As seen from the graph, they are feedforward neural networks.

LSTM (Long Short-Term Memory)

LSTM is an artificial recurrent neural network architecture. Along with the standard feedforward neural networks, LSTM also has feedback connections, making it well-suited for time-series data.



An LSTM cell. Source: Wikipedia

An LSTM unit is composed of a forget gate, an input gate, and an output gate.

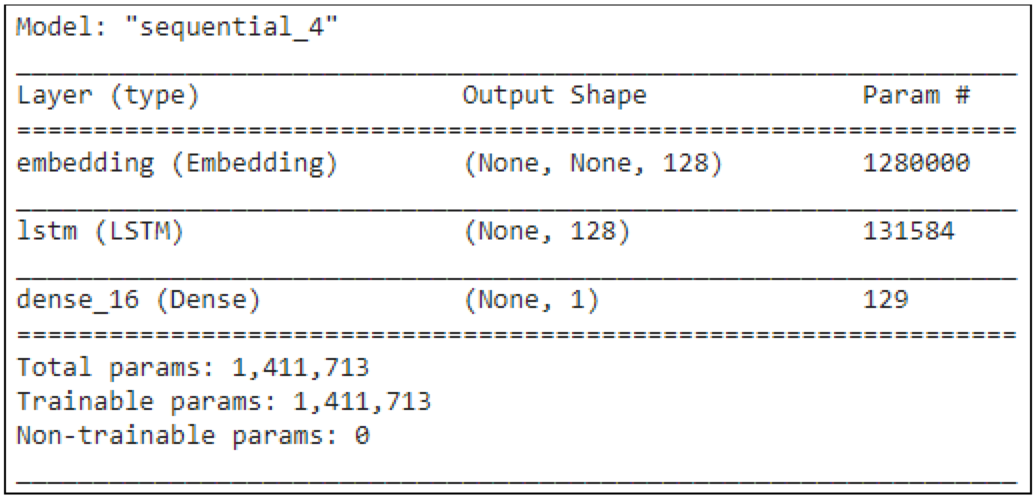
- The first sigmoid function from the left (in orange) represents the forget gate. Its value is either 0 or 1 corresponding to forgetting and remembering respectively.
- The next sigmoid function and tanh function (also in orange) represent the input gate. It controls how much information to pass to the current cell state.

- The bottom line sigmoid function (orange) and the tanh function (in yellow) represent the output gate. It controls how much information is retained in the hidden state.

Building LSTM model

We will use 10,000 unique words for this task from the IMDB dataset. As the Embedding layer and LSTM will be used, we don't need to worry about the memory requirement (as was the case with the matrix representation), but it will take more time to train than the previous model. The data is padded to a maximum length (100) to ensure that the instances have the same length.

The model architecture is as below:



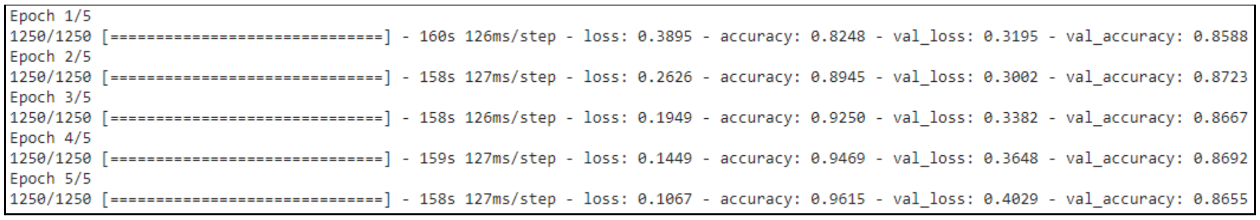
```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, None, 128)         1280000
_________________________________________________________________
lstm (LSTM)                  (None, 128)               131584
_________________________________________________________________
dense_16 (Dense)             (None, 1)                 129
=================================================================
Total params: 1,411,713
Trainable params: 1,411,713
Non-trainable params: 0
_________________________________________________________________
```

The layer is the embedding layer with input_dim = 10,000 (number of unique words) and output_dim = 128. The output is passed to the LSTM layer which does the time series analysis. The last layer is the Dense layer with sigmoid activation to output 0 or 1 (for negative and positive sentiment).

It is expected for this model to take some time to train as it has close to 1.5 million parameters. Let's train with three optimizers to gauge their performances.

Adam Optimizer

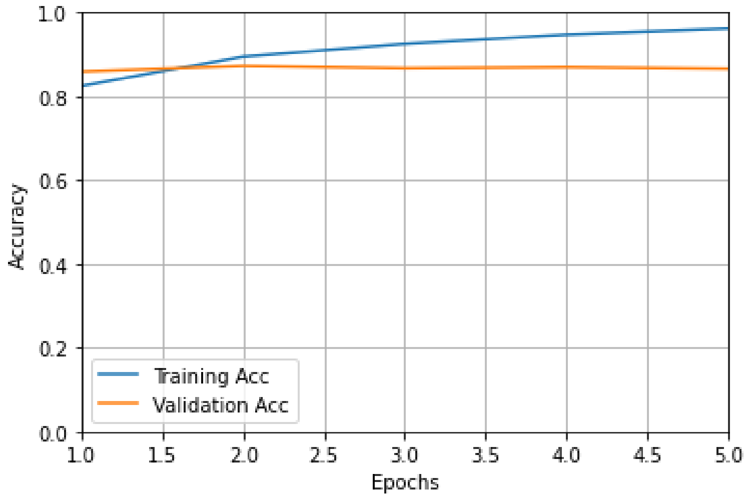
The losses and accuracy on the training set and testing set are:



```
Epoch 1/5
1250/1250 [==============================] - 160s 126ms/step - loss: 0.3895 - accuracy: 0.8248 - val_loss: 0.3195 - val_accuracy: 0.8588
Epoch 2/5
1250/1250 [==============================] - 158s 127ms/step - loss: 0.2626 - accuracy: 0.8945 - val_loss: 0.3002 - val_accuracy: 0.8723
Epoch 3/5
1250/1250 [==============================] - 158s 126ms/step - loss: 0.1949 - accuracy: 0.9250 - val_loss: 0.3382 - val_accuracy: 0.8667
Epoch 4/5
1250/1250 [==============================] - 159s 127ms/step - loss: 0.1449 - accuracy: 0.9469 - val_loss: 0.3648 - val_accuracy: 0.8692
Epoch 5/5
1250/1250 [==============================] - 158s 127ms/step - loss: 0.1067 - accuracy: 0.9615 - val_loss: 0.4029 - val_accuracy: 0.8655
```

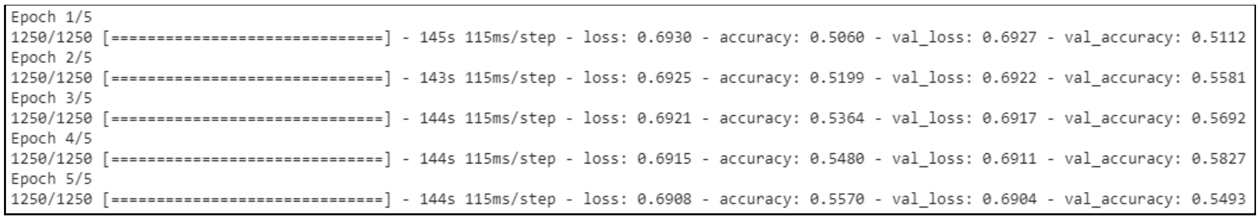
Similar to its performance in MLP, the adam optimizer performs well with over 86% accuracy in the test data. The accuracy for the training set is above 96% (see graph below). There is a separation between the accuracies due to overfitting the training data. Hence, the model needs to be regularized.

Accuracy on Training and Testing set:



SGD optimizer

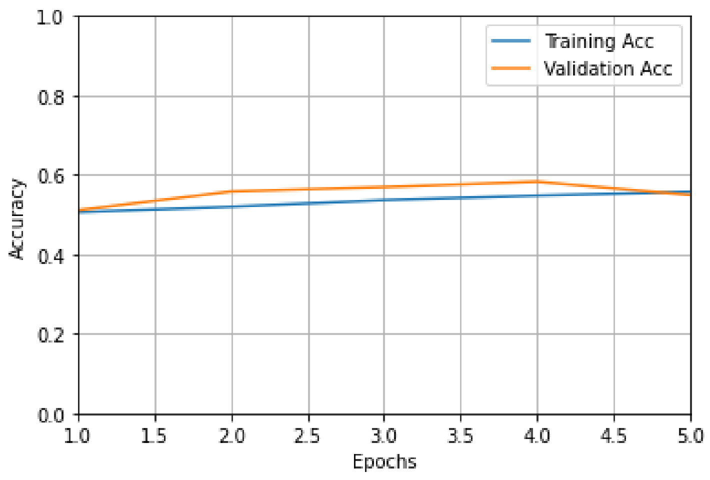
The losses and accuracy on the training set and testing set are:



```
Epoch 1/5
1250/1250 [==============================] - 145s 115ms/step - loss: 0.6930 - accuracy: 0.5060 - val_loss: 0.6927 - val_accuracy: 0.5112
Epoch 2/5
1250/1250 [==============================] - 143s 115ms/step - loss: 0.6925 - accuracy: 0.5199 - val_loss: 0.6922 - val_accuracy: 0.5581
Epoch 3/5
1250/1250 [==============================] - 144s 115ms/step - loss: 0.6921 - accuracy: 0.5364 - val_loss: 0.6917 - val_accuracy: 0.5692
Epoch 4/5
1250/1250 [==============================] - 144s 115ms/step - loss: 0.6915 - accuracy: 0.5480 - val_loss: 0.6911 - val_accuracy: 0.5827
Epoch 5/5
1250/1250 [==============================] - 144s 115ms/step - loss: 0.6908 - accuracy: 0.5570 - val_loss: 0.6904 - val_accuracy: 0.5493
```

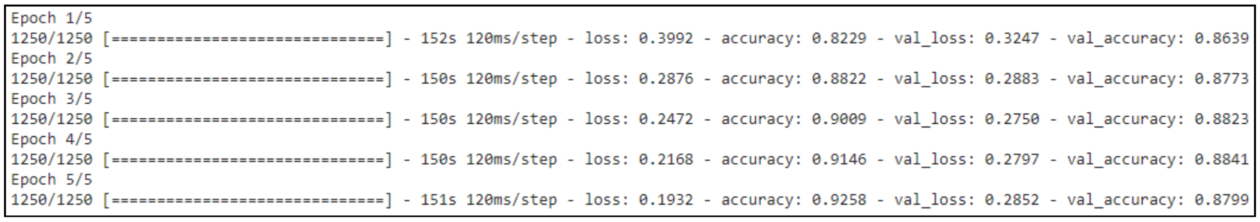
This optimizer performs badly. The accuracy is slightly better than a coin toss. From the below graph, we see that the accuracy is stuck in the 50% range.

Accuracy on Training and Testing set:



RMSprop Optimizer

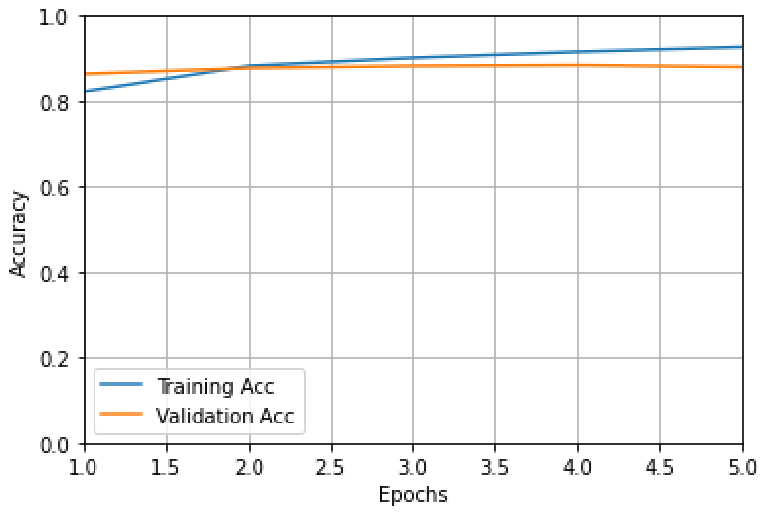
The losses and accuracy on the training set and testing set are:



```
Epoch 1/5
1250/1250 [==============================] - 152s 120ms/step - loss: 0.3992 - accuracy: 0.8229 - val_loss: 0.3247 - val_accuracy: 0.8639
Epoch 2/5
1250/1250 [==============================] - 150s 120ms/step - loss: 0.2876 - accuracy: 0.8822 - val_loss: 0.2883 - val_accuracy: 0.8773
Epoch 3/5
1250/1250 [==============================] - 150s 120ms/step - loss: 0.2472 - accuracy: 0.9009 - val_loss: 0.2750 - val_accuracy: 0.8823
Epoch 4/5
1250/1250 [==============================] - 150s 120ms/step - loss: 0.2168 - accuracy: 0.9146 - val_loss: 0.2797 - val_accuracy: 0.8841
Epoch 5/5
1250/1250 [==============================] - 151s 120ms/step - loss: 0.1932 - accuracy: 0.9258 - val_loss: 0.2852 - val_accuracy: 0.8799
```

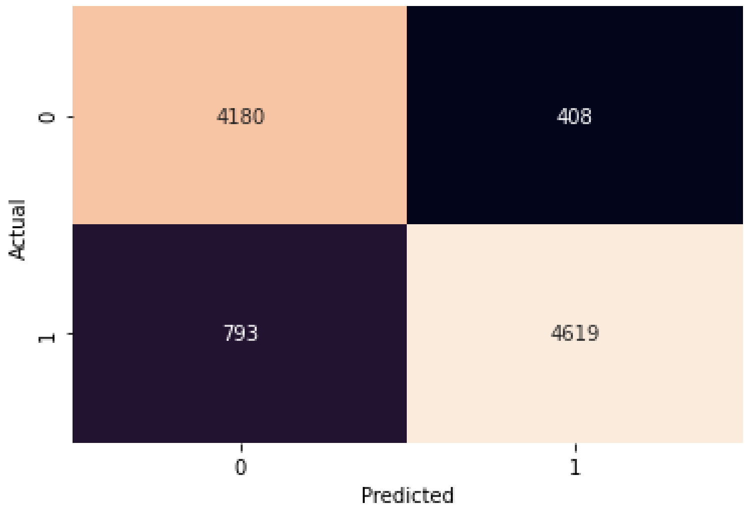
The RMSprop optimizer performs much better than the other optimizers. The accuracy of the testing set is ~ 88% and that of the training set is ~ 92%. The separation in the below graph is very small. Hence this model performs better than the previous ones without severely overfitting the data.

Accuracy on Training and Testing set:



Confusion Matrix & mislabelled class

From the confusion matrix below, we see that from 10,000 test instances, about ~ 8800 were correctly labeled. That is over 88% accuracy.



More positive reviews were mislabeled as negative (false negative) than the negatives as positive (false positive). Looking at the mislabeled instances:

False Negative reviews:

- marketing plan to **destroy** anything that looks remotely original and promising br br conclusion you have a show with good special effects stuff like huge monsters **killing people or destroying** # then going into genetic # # people human # end of the world also the only # are scenes with aggressive # or other **annoying people** being killed for their **stupidity** the down side is that after 15 episodes that prepare something huge the show ends **no real ending no closure** just a bitter taste of cloth in one's mouth as if you just # a piece of suit

- show is how original it is and the fact that it has people the # who love doing what they do and who would # this show has loads of **bits** in it that can **crack up** anybody like the ad road test which i've already mentioned mr ten questions where he asks 10 questions really fast in front of famous celebrities like hugh jackman and the beach boys **temporary** ones like the # # and ones that have been there from the start like what have we # from current affairs this week overall i rate **thin** show 98

- that there are **strange goings on** in the corridors at night and there are even **stranger** events taking place out back in hollywood's most familiar # # yes this is the # you've seen in every feature length film that ever involved a # puzzle up to and including the shining **but** the punchline to this story is about the last thing you would guess i certainly didn't see it coming this is a fine example of how good and convincing a movie can be even when the premise is **utterly** # # on laughable i'd recommend it to anyone

False Positive reviews:

- little bit of slapstick relevant joe de rita despite his background in vaudeville is just not up to the job as a replacement for curly # or even joe # if that's who moe and larry had left to pick from they should have just closed up shop and **enjoy** their retirement years leaving us fans with **better memories** of far **better films** they had done earlier always leave them **laughing** is the # for **comedy** and always quit while you're on **top** hence # leaving the sitcom while right up there instead of sticking around for the inevitable decline

- africa then go to to pick up fish in order not to make the trip back empty meaning that they do actually land empty in br br people do eat fish # contrary to what the film suggest around 40 60 of what is taken out of the lake and thousands of people make their living with it **good** for them it's private business of that kind that will one day take african countries out of poverty and not western # and endless foreign **assistance** br br i cannot tell how shocked i am seeing the **success** of this film

- especially # ones that may view sandra as a hero there is a scene where the writers producers directors thought it would be nice to show how acceptable it is to smoke a joint while driving and then have no consequences at all when caught i'm no # and i # my share when i was younger but i **guarantee** you i won't take my teenagers to see it and they're solid a b students if you want to see a **good** sandra # movie rent the net or hope # which i believe are two of her **best works**

Bolding out some of the words that might have given the model false signals. We see that it is indeed hard to get the sentiment of these reviews.

Note that “br” means newline, “#” means the embedded integer wasn’t in the 10,000 unique words which we selected in the beginning.

What More?

- More training data: Taking more unique words as input to the embedder will give better results, although it will increase the training time.
- Data Augmentation: A collection of negative and positive sentences can be created. Appending different sentences into one review will create an instance for the training data. This technique has potential but can also skew the data or bias the output badly.
- Training a more complex model: A more deep and complex model can be built to improve the performance further.
- Other Optimizers: There are other optimizers that can be explored, like Nadam, Adamax, Ftrl, etc. Custom models can also be built according to the requirements.

References

https://searchenterpriseai.techtarget.com/definition/recurrent-neural-networks
https://en.wikipedia.org/wiki/Long_short-term_memory