Discussion:

1.

In this task we will create a simple RNN model to classify sentiments in binary format. We will be using IMDB dataset and Keras package.

2.

In this step, we implemented and trained RNN for sentimental analysis using Keras. We were required to use an IMDB dataset to execute this task. The training data consists of 25,000 highly polar movie reviews and 25,000 testing data. The given problem at hand is to analyze if the movie given is positive or negative. The reviews contain word IDs and are stored in the form of integers.

3.

Here, the raw dataset underwent the process of being tokenized and padded. This is essential so that the input data to the RNN are all kept the same length. If the reviews are too short, they undergo padding by adding a value of zero. This is implemented using the pad_sequences () function. On the other hand, if the reviews are too long, they go through a parameter known as “max_words”. This helps in limiting the word count.

4.

RNN is nothing but abstract concept of sequential memory. In this final output is created using whole sequence of data which is then forwarded to feed forward network which then classifies the output. RNNs usually have a very short-term memory which is due to its very famous vanishing gradient problem. In our case we are building a simple RNN model which has an embedding layer in the beginning followed by an RNN layer and followed by a fully connected layer which then goes to the output classifier using a sigmoid function.

5.

a)

Long short-term memory (LSTM): It a sub-type of RNN which mitigates the problem of short-term memory. They use the concept of gates to reduce the vanishing gradient problem. Gates work like memory blocks which controls the information flow through the network. The gates have work very much like an editable document where we can block or pass the information which are filtered based on their weights.

b)

Gated Recurrent Units (GRU): GRUs are improved version of standard recurrent neural network. To solve the vanishing gradient problem of a standard RNN, GRU uses, so-called, update gate and reset gate. GRUs are able to store and filter the information using their update and reset gates. since the model is not washing out the new input every single time but keeps the relevant information and passes it down to the next time steps of the network.

c)

Bidirectional LSTM: This network performs better for a classification involving sequential data. The model is trained using two LSTMs- the original one being the first one and reversed version of the original sequence as the second one. The idea is to split the state neurons of a regular RNN in a part that is responsible for the positive time direction (forward states) and a part for the negative time direction (backward states).

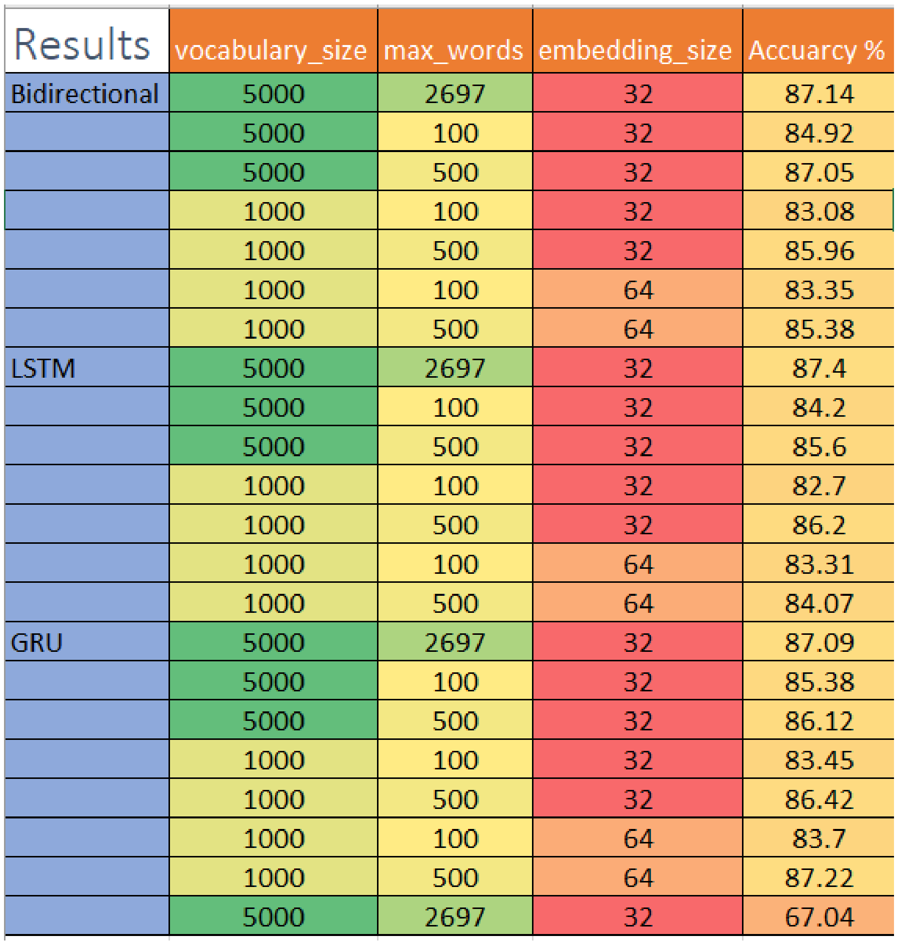
6.

The performance variation can be seen in the results section below.

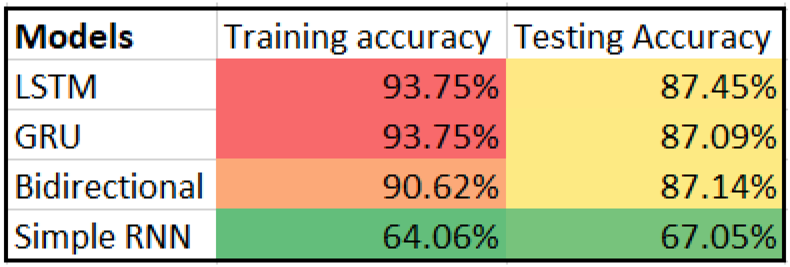
Results:

In the above given experiment, we ran total 4 models Simple RNN, LSTM, GRU, Bidirectional RNN. As our instinct goes the worst performance was displayed by a Simple RNN they have a vanishing gradient issue the accuracy of that model was 67.04% with training accuracy as 64.06%

While for all the other models I ran the code with different combinates by varying the Max length as 2697, 500 and 100 while the values of Embedding size were varied between 32 and 64 and values of vocabulary size were varied between 5000, 1000.



Results	vocabulary_size	max_words	embedding_size	Accuarcy %
Bidirectional	5000	2697	32	87.14
	5000	100	32	84.92
	5000	500	32	87.05
	1000	100	32	83.08
	1000	500	32	85.96
	1000	100	64	83.35
	1000	500	64	85.38
LSTM	5000	2697	32	87.4
	5000	100	32	84.2
	5000	500	32	85.6
	1000	100	32	82.7
	1000	500	32	86.2
	1000	100	64	83.31
	1000	500	64	84.07
GRU	5000	2697	32	87.09
	5000	100	32	85.38
	5000	500	32	86.12
	1000	100	32	83.45
	1000	500	32	86.42
	1000	100	64	83.7
	1000	500	64	87.22
	5000	2697	32	67.04



Models	Training accuracy	Testing Accuracy
LSTM	93.75%	87.45%
GRU	93.75%	87.09%
Bidirectional	90.62%	87.14%
Simple RNN	64.06%	67.05%

As we can see the best performance was given by LSTM model which was followed by GRU an in the end by Bidirectional model. Also, the results varied depending on the max words used, vocabulary size and embedded size as we can clearly observe as the value of max size reduces the accuracy of model drops as it tends to underfit the data. With reduction in vocabulary size the accuracy drops and the same happens by reducing the size of embedded size. Although we cannot even blindly keep increasing the values of these parameters to increase the accuracy as the model may start to overfit and start giving even worse results.

I have also attached .pdf files of every run main .py files for all 4 models for the reference as an attachment.