Assignment 4: Text and Sequence

By: Harika Beesetti and Arvind Chaurasia

Objective:

The IMDB dataset poses a binary classification challenge: sorting movie reviews into positive and negative categories. With 50000 reviews in the dataset, only the top 10,000 words are considered. Training samples vary in size, ranging from 100 to 100,000, while validation is conducted on a fixed set of 10,000 samples. After data preparation, it undergoes processing in the embedding layer with the aid of a pre-trained embedding model. Various strategies are then explored to gauge their effectiveness in improving performance.

In the preprocessing stage, each review transforms into a set of word embeddings, where individual words are represented by fixed-size vectors. However, there is a cap of 10,000 samples in this process. Additionally, a numerical series corresponding to each word is created from the reviews, replacing the string format. Although the neural network's input can't directly accommodate this numerical sequence.

To prepare the data for the neural network, tensors are constructed. These tensors are generated from the integer lists, adopting an integer data type and structured as (samples, and word indices). Ensuring uniformity across samples, each review's length is standardized by padding with dummy words or numbers. This guarantees consistent input dimensions for the network.

Problem Statement:

The purpose of this assignment is to apply RNNs, or Transformers to text and sequence data. module learning outcomes aimed at understanding and implementing these architectures effectively.

Further, we will enhance network performance, particularly in scenarios with limited data availability. We will also determine the most suitable approaches for prediction improvement.

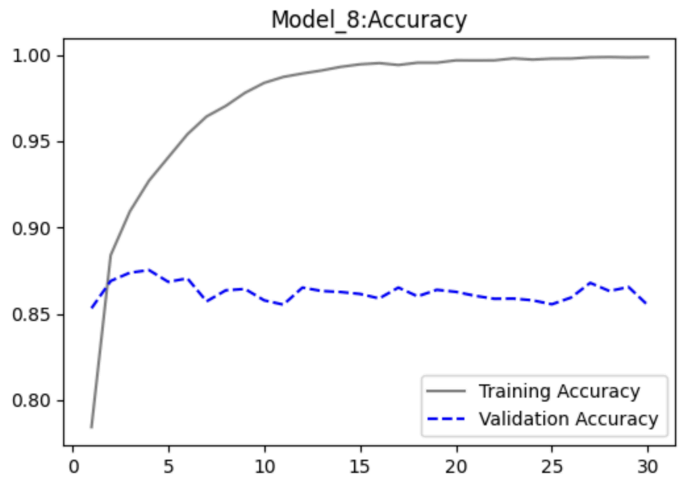
Results:

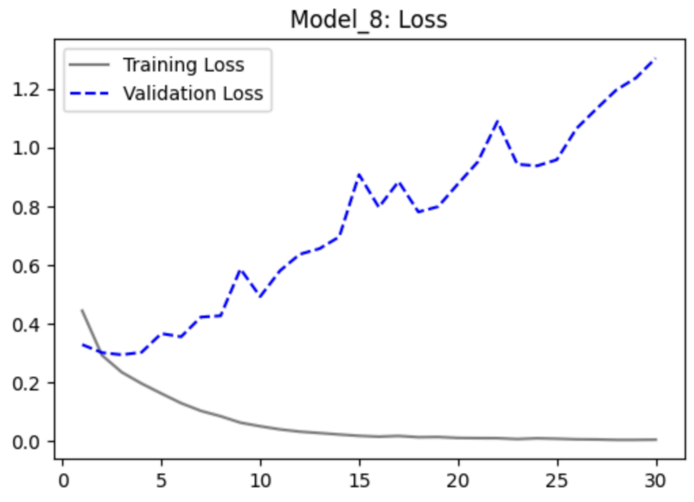
Model No.	Model Description	Test Accuracy
1	Basic model just uses an embedded layer with a Training Sample size of 100	50%
2	Basic model just uses an embedded layer with a training sample of 5000	85.8%
3	Basic model just uses an embedded layer with a training sample size of 10000	88.2%
4	Using convolution 1D and Embedding layer together with a training sample size of 10000	83.7%
5	A sequence model built on one-hot encoded vector sequences with LSTM	87.9%
6	LSTM using an embedded layer with Training Sample 10000	86.4%
7	LSTM using an embedded layer with Training Sample 20000	88.8%
8	LSTM with embedding layer and Masking enable with training sample size as 20000	90.2%
9	Transformers with embedding layer and training sample size 20000	88.8%
10	Pretrained Models with Training sample size 100 we are using the GloVe model	49.5%
11	Pretrained Models with 4 LSTM hidden layers with Training sample size 5000	66.2%
12	Pretrained Models with 2 LSTM hidden layer with Training sample size 15000	54.9%

<u>Insight & Observations:</u>

In the case of simple models with one embedded layer, we tested a simple model in three ways, using different amounts of training data. The more data we used, the better the model performed. For example, with only 100 examples, the model wasn't very good (50% accurate). But with 10000 examples, the accuracy increased to 88.2%.

When we used the LSTM model we achieved a significantly higher test accuracy in nearly all the models. LSTM using an embedded layer and training sample of 20000 achieved 90.2% accuracy. Similarly, a transformer with an embedding layer also performed well.





The above graph shows the training and validation accuracy over the number of epochs. The training and validation loss is also shown over the 30 epochs. This is the graph for LSTM with embedding layer and Masking enabled with training sample size as 20000.

Pretrained models were also leveraged with and without LSTM hidden layers. Their performance is mentioned in the table. We observed that the accuracy decreased to 54.9% when we reduced the number of LSTM layers and increased the sample size from 5000 to 15000.

All the graphs related to the training, validation accuracy, and related losses can be seen in the code.

Conclusion:

RNNs are sequential models with recurrent connections that process input sequentially, while the Transformer Architecture is based on self-attention mechanisms and operates on entire sequences in parallel, making it more efficient for capturing long-range dependencies in sequences like natural language.

The LSTM model has long and short-term memory which is very helpful for handling sequence problems, and has shown a good testing accuracy for all its models.