Ahmed Mohamed

Cpts 437

Final Project

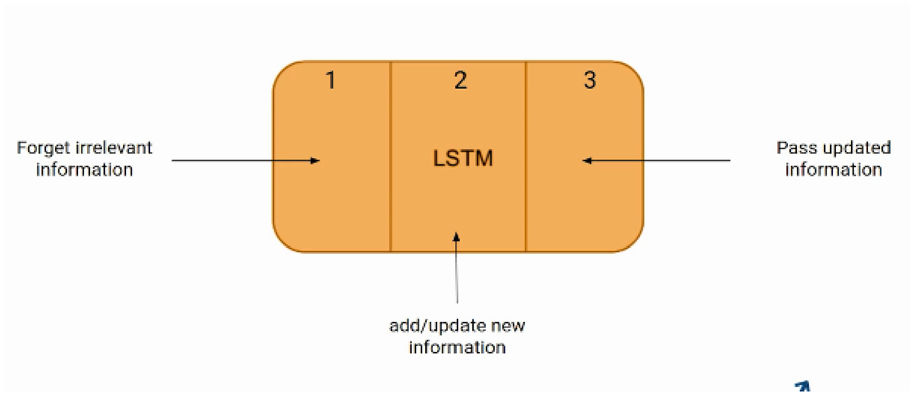
Title: LTSM to perform sentiment analysis on IMDB Dataset

Introduction

This project aims to implement one of the most successful RNN architecture for sequence. LTSM model was one of the first successful techniques used for addressing vanishing gradients. LTSM resembles recurrent neural networks, however, each recurrent node is replaced by a memory cell. Each cell contains a node with self-connected recurrent edges of weight value of 1. This is to ensure that the gradient can pass through many times without vanishing or exploding. The intuition behind Long short-term memory is simple recurrent neural networks have long term memory in form of weight. The weight changes during training to encode general knowledge about the data. In the other hand, short-term memory in the form of ephemeral activation which passes between each node and its successive nodes. LTSM model memory cell is a composite unit built from simpler nodes in a connected pattern with novel inclusion of multiplicative nodes. LTSM excels in capturing long-term dependencies which in retrospect makes it good for predicting sequences. Contrary to traditional neural networks, LTSM incorporates feedback connections to allow for processing entire sequence of data not just individual data points. This will help in terms of effectiveness regarding predicting patterns in data such as text, which the movie data set is based on.

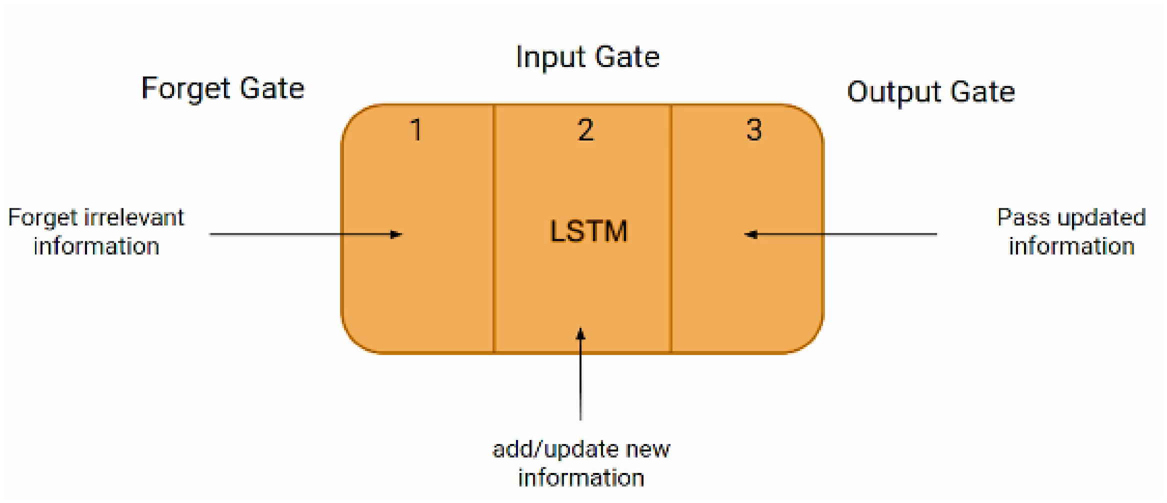
LTSM Architecture

This section will discuss how LTSM resolves the vanishing gradient problem that is common in RNN models. At a higher level, LTSM works very similar to an RNN cell. LTSM network architecture consists of three parts performing individual functions.

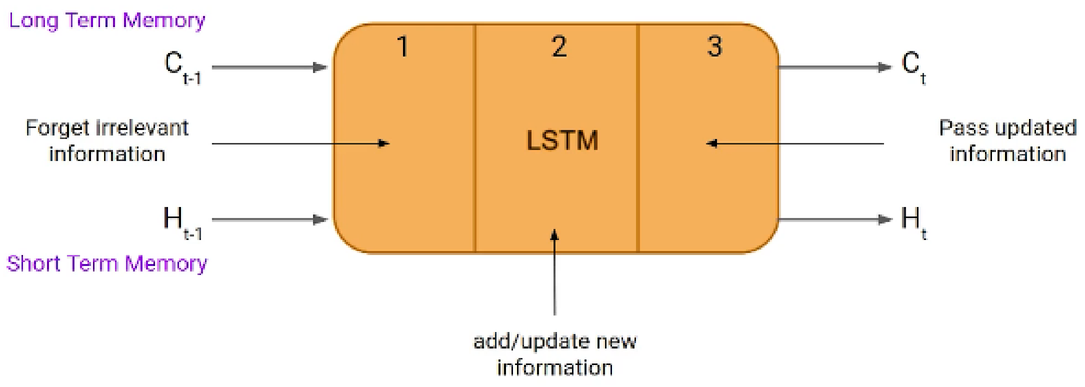


The Logic behind LTSM is that first part chooses whether incoming information from previous time stamp is to be remembered or forgotten. The second part, the cell tries to learn new information from the input to the cell. Finally, the cell passes the updated information from the current time stamp to the next timestamp. This process is considered one cycle of LTSM or a single time step.

The parts identified above are identified as "gates." As mentioned, the data feed into the LTSM gates are the input at the current time step and the hidden state of the previous time step. All three layers with sigmoid activation functions compute the values of the input, forget and output gates. The sigmoid activation results in values of the three gates being in ranges from 0 to 1. In addition, an input node is typically computed with tanh activation function. Which determines how much of the input node value should be added to the current memory cell internal state. The forget gate decides whether to keep the current value of the memory or flush it. Finally, the output gate determines whether memory cell should influence the output at the current time step. These three gates and a memory cell or "LTSM cell" can be considered a layer of neurons in the since of traditional feedforward neural networks where each neuron has a hidden layer and a current state.



The figure below shows the hidden state (short term memory) and cell state (long term memory)



Mathematical Theory

Mathematically, suppose that there are h hidden units, the batch size is n, and the number of inputs is d. Thus, the input is Xt ∈ ℝ^(nxd) and the hidden state of the previous time step is Ht-1 ∈ ℝ^(nxh). Correspondingly, the gates at time step t are defined as follows: the input gate is It ∈ ℝ^(nxh), the forget gate is Ft ∈ ℝ^(nxh), and the output gate is Ot ∈ ℝ^(nxh). They are calculated as follows:

$$It = \sigma(XtWxi + Ht\text{-}1Whi + bi),$$

$$Ft = \sigma(XtWxf + Ht\text{-}1Whf + bf),$$

$$Ot = \sigma(XtWxo + Ht\text{-}1Who + bo),$$

where Wxi, Wxf, Wxo ∈ $\mathbb{R}$^(dxh) and Whi, Whf, Who ∈ $\mathbb{R}$^(hxh) are weight parameters and bi, bf, bo ∈ $\mathbb{R}$^(1xh) are bias parameters. Note that broadcasting (see Section 2.1.4) is triggered during the summation. We use sigmoid functions (as introduced in Section 5.1) to map the input values to the interval (0, 1).

Input Node

the input node Ĉt ∈ $\mathbb{R}$^(nxh). Its computation is similar to that of the three gates described above but uses a tanh function with a value range for (−1, 1) as the activation function. This leads to the following equation at time step t:

$$\hat{C}t = \tanh(XtWxc + Ht\text{-}1Whc + bc),$$

where Wxc ∈ $\mathbb{R}$^(dxh) and Whc ∈ $\mathbb{R}$^(hxh) are weight parameters and bc ∈ $\mathbb{R}$^(1xh) is a bias parameter.

Memory Cell Internal State

In LSTMs, the input gate It governs how much we take new data into account via Ĉt, and the forget gate Ft addresses how much of the old cell internal state Ct−1 ∈ $\mathbb{R}$^(nxh) we retain. Using the Hadamard product operator ⊙ we arrive at the following update equation:

$$Ct = Ft \odot Ct{-}1 + It \odot \hat{C}t.$$

If the forget gate consistently has a value of 1 and the input gate is fixed at 0, the internal state of the memory cell from the previous time step remains unaltered across all future steps. This means the cell's memory is static and doesn't update. However, the ability of input and forget gates to adjust their values enables the model to decide dynamically whether to maintain or modify its memory. This flexibility is important for the model to effectively handle new information and avoid issues like the vanishing gradient problem, which can make training challenging. Such a mechanism is beneficial for learning from data with long sequences, making the model training process more manageable.

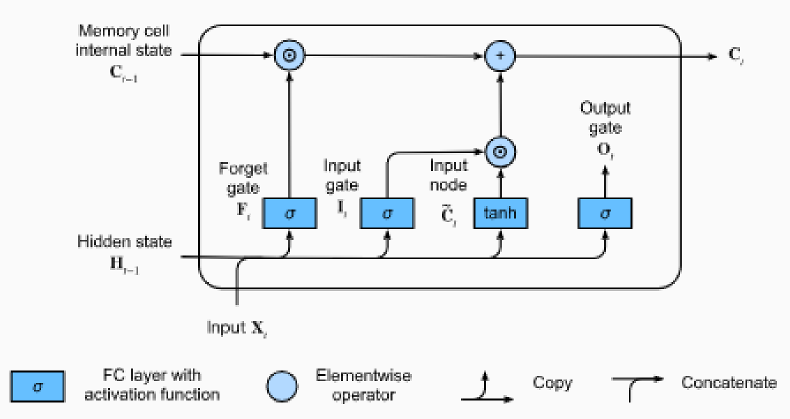

Memory cell internal state $\mathbf{C}_{t-1}$
$\mathbf{C}_t$
Output gate $\mathbf{O}_t$
Forget gate $\mathbf{F}_t$
Input gate $\mathbf{I}_t$
Input node $\tilde{\mathbf{C}}_t$
σ
tanh
Hidden state $\mathbf{H}_{t-1}$
Input $\mathbf{X}_t$
FC layer with activation function
Elementwise operator
Copy
Concatenate

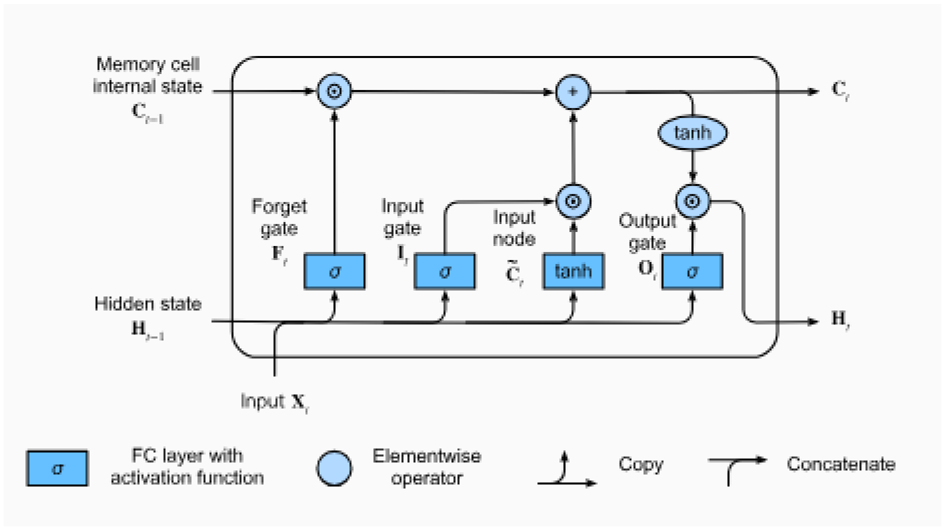
Fig. 10.1.3 Computing the memory cell internal state in an LSTM model.

Hidden State

To determine the output from a memory cell in LSTMs, or its hidden state Ht ∈ ℝ^(nxh), which is observable to other network layers, we employ the output gate. The process involves transforming the internal state of the cell using the hyperbolic tangent function (tanh) to keep the output values within the range of -1 to 1. This transformation is represented mathematically as:

Ht = Ot ⊙ tanh(Ct).

The output gate's value dictates how much the cell's state affects the network. A value near 1 means the cell's state freely influences subsequent layers, while a value near 0 blocks any immediate effect on the network. This allows the cell to store information over multiple time steps without altering the network's behavior, only exerting an influence when the output gate's value increases to near 1. This mechanism helps manage the flow of information and is graphically depicted in the referenced figure.


Memory cell internal state $\mathbf{C}_{t-1}$
Forget gate $\mathbf{F}_t$
Input gate $\mathbf{I}_t$
Input node $\tilde{\mathbf{C}}_t$
Output gate $\mathbf{O}_t$
tanh
σ
Hidden state $\mathbf{H}_{t-1}$
Input $\mathbf{X}_t$
$\mathbf{C}_t$
$\mathbf{H}_t$
FC layer with activation function
Elementwise operator
Copy
Concatenate

Application

Environment Setup: Pandas, NumPy, Matplotlib, Seaborn, NLTK, and Keras.

Configuration: LSTM with 100 units per layer is used. The embedding layer has an output dimension of 100.

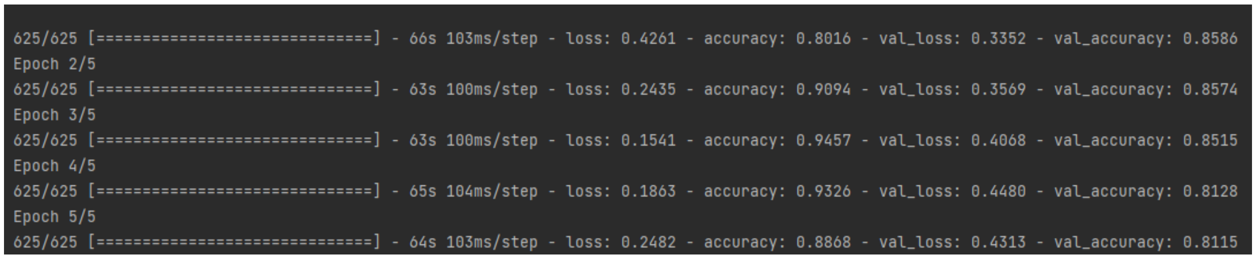
Database: https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews

Installation:

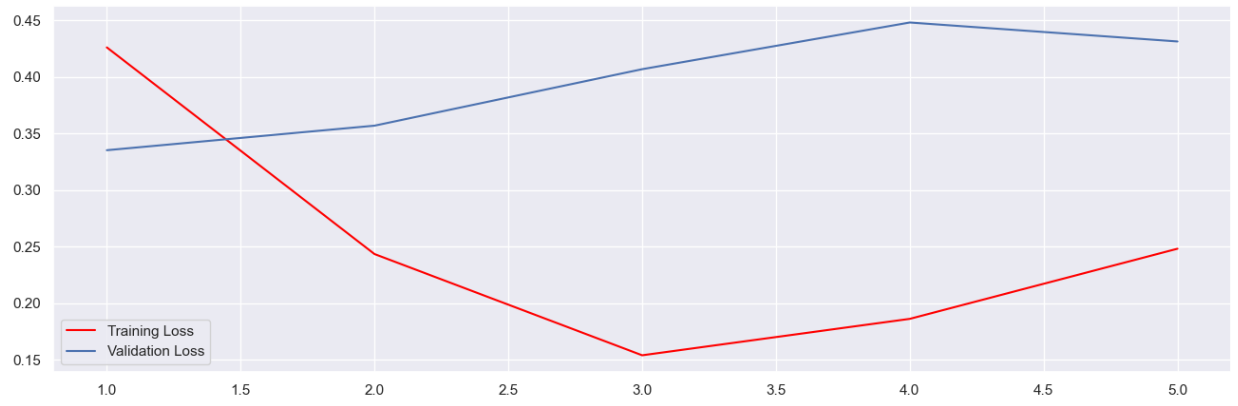
- pip install pandas numpy matplotlib seaborn nltk keras

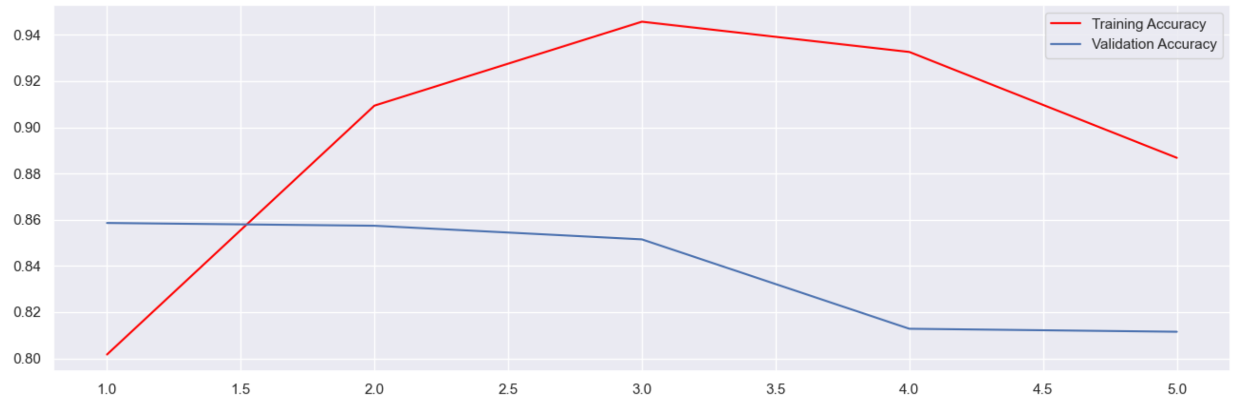
Results:

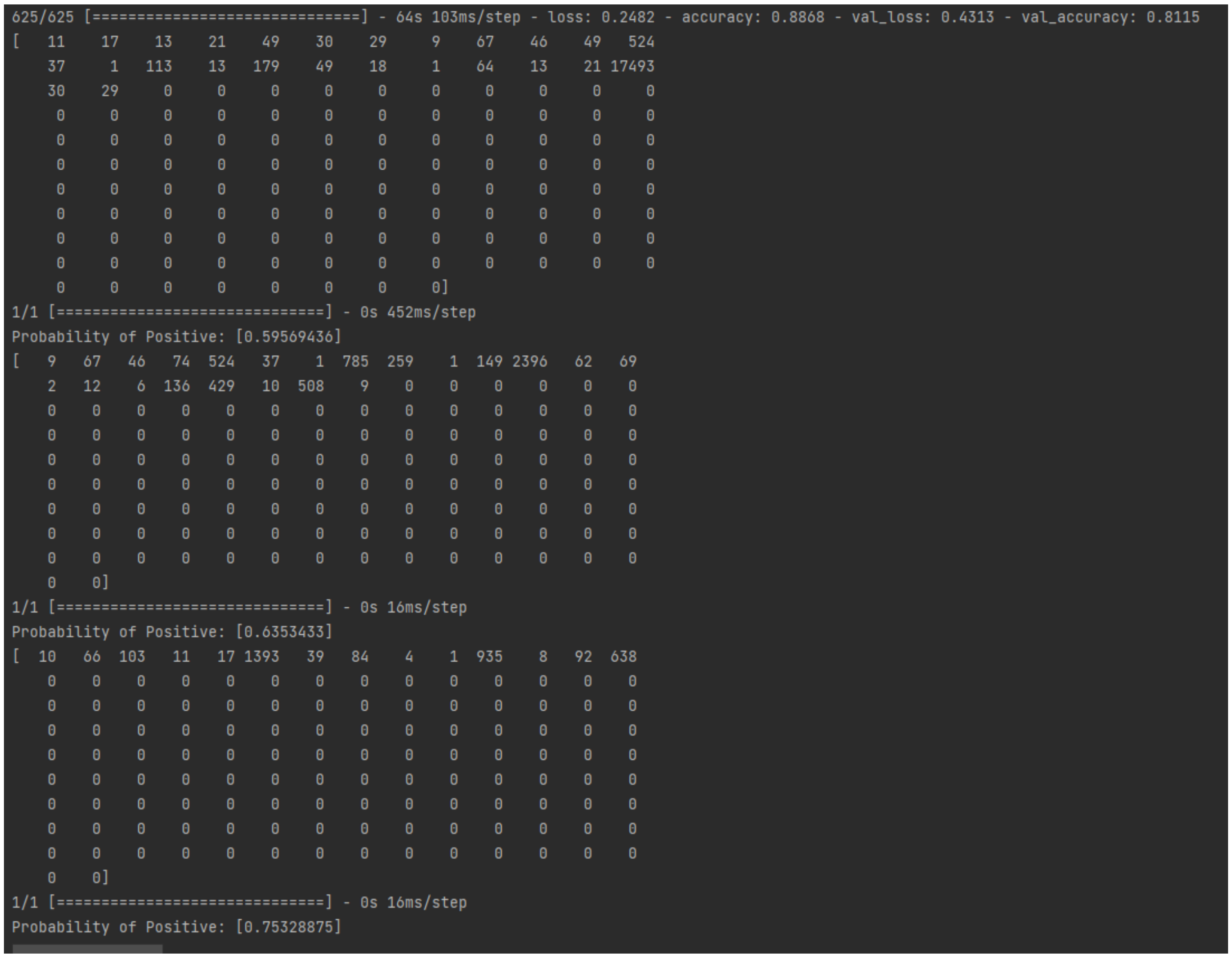
After running only 5 Epochs:



```
625/625 [============================] - 66s 103ms/step - loss: 0.4261 - accuracy: 0.8016 - val_loss: 0.3352 - val_accuracy: 0.8586
Epoch 2/5
625/625 [============================] - 63s 100ms/step - loss: 0.2435 - accuracy: 0.9094 - val_loss: 0.3569 - val_accuracy: 0.8574
Epoch 3/5
625/625 [============================] - 63s 100ms/step - loss: 0.1541 - accuracy: 0.9457 - val_loss: 0.4068 - val_accuracy: 0.8515
Epoch 4/5
625/625 [============================] - 65s 104ms/step - loss: 0.1863 - accuracy: 0.9326 - val_loss: 0.4480 - val_accuracy: 0.8128
Epoch 5/5
625/625 [============================] - 64s 103ms/step - loss: 0.2482 - accuracy: 0.8868 - val_loss: 0.4313 - val_accuracy: 0.8115
```







```
625/625 [==============================] - 64s 103ms/step - loss: 0.2482 - accuracy: 0.8868 - val_loss: 0.4313 - val_accuracy: 0.8115
[   11    17    13    21    49    30    29     9    67    46    49   524
    37     1   113    13   179    49    18     1    64    13    21 17493
    30    29     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0]
1/1 [==============================] - 0s 452ms/step
Probability of Positive: [0.59569436]
[   9   67   46   74  524   37    1  785  259    1  149 2396   62   69
    2   12    6  136  429   10  508    9    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0]
1/1 [==============================] - 0s 16ms/step
Probability of Positive: [0.6353433]
[  10   66  103   11   17 1393   39   84    4    1  935    8   92  638
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0]
1/1 [==============================] - 0s 16ms/step
Probability of Positive: [0.75328875]
```

Conclusion

The LTSM model applied to the sentiment analysis of the IMDB database achieved decent accuracy, peaking at 88% on the training set and 91.15% on the validations set after only 5 epochs. The training trajectory is seen by the consistent reduction in loss and peak accuracy implies the model capacity to learn language patterns. On the other hand, the slight increase in the validation loss indicates potential overfitting which suggests caution when optimizing generalization. The model prediction on individual reviews displays it efficacy in understanding sentiment.

Sources:

https://d2l.ai/chapter_recurrent-modern/lstm.html

https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/

https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews

https://www.youtube.com/watch?v=yK9Ya6KzVAg

https://www.youtube.com/watch?v=Nk8nM2anJTw

https://www.youtube.com/watch?v=oWo9SNcyxlI&t=337s