Text and Sequence

Objective :
To build a model by applying RNN to text and sequence data using IMDB data set to determine a positive or negative review and understanding the factors improving model performance when dealing with limited data and determining best approaches for suitable prediction

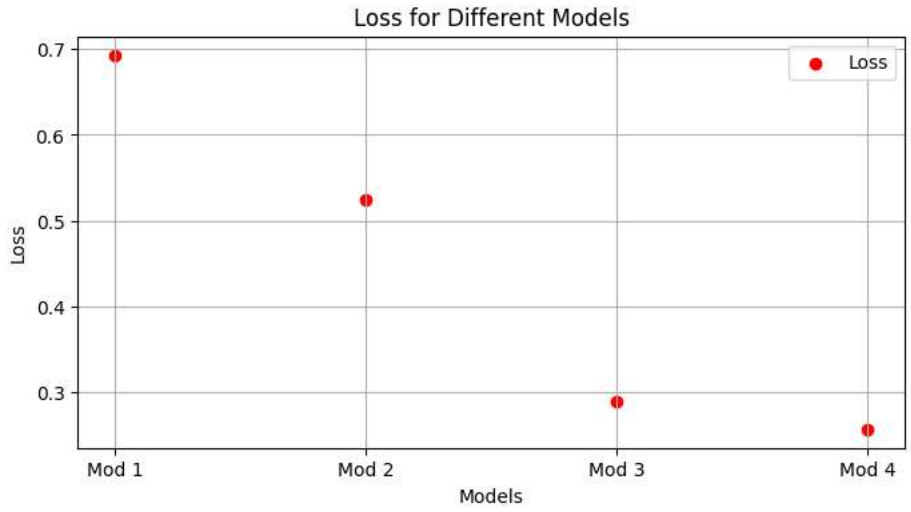
Models:

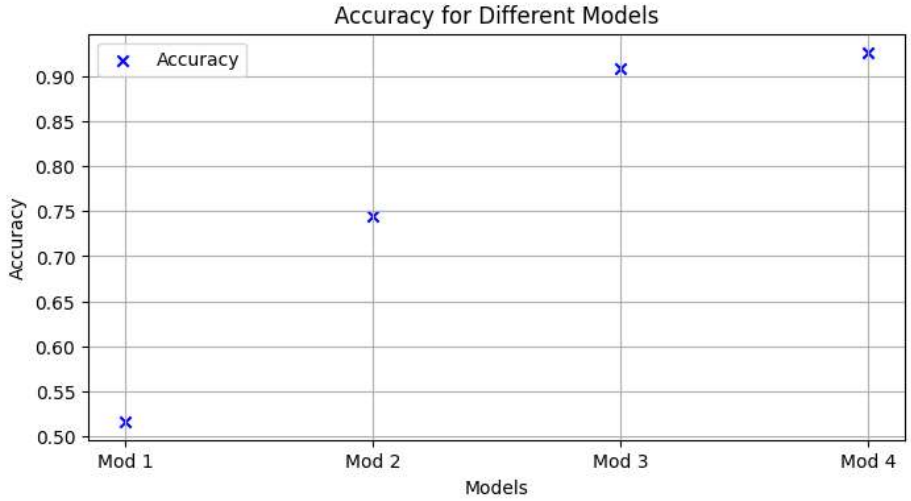
Model 1: This model underwent training on a relatively small dataset of 100 samples, with validation occurring on 10,000 samples. The final evaluation, conducted on a test set of 5,000 samples, resulted in a test loss of 0.693 and a test accuracy of 0.498.

Model 2: With a larger training dataset of 1,000 samples, Model 3 demonstrated improved performance compared to Model 2. Validation was conducted on 10,000 samples, and the test set, comprising 5,000 samples, yielded a lower test loss of 0.524 and a higher test accuracy of 0.745.

Model 3: This model was trained on a substantial dataset of 25,000 samples, with validation and test sets similar to Models 2 and 3. Model 4 showcased notable improvement, achieving a lower test loss of 0.312 and a higher test accuracy of 0.900.

Model 4 :Among the models, Model 4 had the largest training dataset, consisting of 35,000 samples. Validation and test sets remained the same. Model 5 outperformed its predecessors, boasting the lowest test loss of 0.258 and the highest test accuracy of 0.925. Overall, there is a discernible trend of increasing performance with the enlargement of the training dataset from Model 1 to Model 4.





The first two models, Model 1 and Model 2, started with a small set of data and a basic structure using an embedding layer. They did okay but weren't outstanding. Moving to Model 3 and Model 4, we made things a bit complex by adding Convolutional 1D layers along with embedding and dense layers. This change taught us a couple of things: first, having more training examples helped the model understand positive and negative aspects in reviews better, making it perform better on the test set. Second, adding Conv1D with the embedding layer made the model stronger, learning from reviews more systematically.

Fine-tuning hyperparameters was important, making adjustments to things like embedding size, learning rates, Conv1D and dense layers, dropout rates, and nodes. This fine-tuning aimed to make the model better at understanding things.

Using visuals, like charts showing training and validation results, was super helpful. It helped us find the best point before the model started getting too specialized. Our strategy included letting

the model get too specialized at first and then fixing things, a method that worked well in making the model better at generalizing, especially in a complex structure.

A big lesson was the idea of giving the model more data when it doesn't do well, which turned out to be a smart move.

In the end, the model 4 took the spotlight. Trained with 35,000 samples, validated with 1,000 samples, and tested with 5,000 samples, this model turned out to be the best. Tweaked hyperparameters and the use of Conv1D layers made it really good at understanding things, marking it as the best result from our step-by-step model-building process.

Pretrained_Model1:
Pretrained_Model1 was trained on a minimal dataset of 100 samples and validated on 10,000 samples. The test set, comprising 5,000 samples, resulted in a test loss of 0.690 and a test accuracy of 0.699. The model's performance, while not exceptional, provides a baseline for comparison.

Pretrained_Model2:
With a more substantial training dataset of 1,000 samples, Pretrained_Model2 demonstrated improved performance. Validation on 10,000 samples and testing on 5,000 samples yielded a lower test loss of 0.614 and a higher test accuracy of 0.713. This suggests that increasing the training data had a positive impact on the model's predictive capabilities.

Pretrained_Model3:
Pretrained_Model3 was trained on 10,000 samples, but its test performance was less impressive. The test loss was 0.691 with a test accuracy of 0.585. Despite the larger training dataset, this model's generalization on the test set appears to be challenging.

Pretrained_Model4:
Trained on 15,000 samples, Pretrained_Model4's performance on the test set was suboptimal, with a test loss of 0.696 and a test accuracy of 0.505. This suggests potential issues with model complexity that need to be addressed.

From building these models, i understand that underfitting is a more significant concern than overfitting. Underfitting indicates poor performance on the training set, suggesting even worse results on unseen data. Pre trained Model 3 and pre trained Model 4 fall into the category of underfit models, as they didn't achieve satisfactory performance during training, and their test set performance was even worse.

Surprisingly, the notion that more complex architectures always lead to better models is challenged.

Among the four models the best was the 2nd pre trained Model. The model, built with 1000 training samples, 10,000 validation samples, and 5,000 test samples, emerged as the best.