Analysis

Link to repo : https://github.com/nsumesh/641HW3
Hardware Used is Apple M3 on CPU with a 16GB RAM

Dataset Summary

The dataset used for the comparative analysis of different RNN architectures for sentiment analysis is the IMDB movie reviews dataset. It contains 50,000 reviews associated with its binary sentiment (0 for negative, 1 for positive) which is evenly split into 25,000 training and 25,000 testing samples

This entire dataset was preprocessed with various steps taken to clean each review. The steps taken were:

- Text cleaning : All reviews were converted to lowercase and punctuation marks, numbers, and special characters were removed to normalize the input.
- Tokenization : The cleaned text was tokenized using the **Keras Tokenizer** which builds an integer index based on the word frequency. The tokenizer was limited to holding the top 10,000 words to reduce sparsity and to hold majority of the content in the reviews
- Truncation of input lengths : Each review was then represented as a series of integer tokens. All sequences were truncated to fixed lengths of 25, 50 and 100 tokens for different configurations.
- This data was then stored separately as csv files containing the training and testing data for each of the 3 sequence lengths

Model Configuration

The sentiment classification experiments were conducted on three different recurrent neural network architectures - RNN, LSTM and Bidirectional LSTM which was implemented using Tensorflow. The common architectural parameters have been listed below.

- Embedding dimension : 128, each word is mapped to a 128 dimensional dense vector, enabling the network to learn semantic representations
- Hidden Size: 128, This is the number of neurons in each recurrent layer
- Number of layers : 1 recurrent layer and 1 output layer
- Dropout : 0.4, It is applied on the recurrent layer to prevent overfitting
- Loss Function : Binary Cross Entropy, appropriate for binary sentiment classification tasks

The optimizers and training settings used for the experiments are:

- Optimizers : Adam, SGD, RMSprop
- Activation Functions : ReLU, Tanh, Sigmoid
- Gradient Clipping : Enabled or Disabled
- Batch Size : 32
- Epochs : 10
- Learning rate : Default for every optimizer

Each architecture (RNN, LSTM, BiLSTM) was defined in separate builder functions (build_rnn, build_lstm, build_bilstm) within models.py. The model configuration remained consistent across experiments except for the sequence length, activation, optimizer and gradient clipping settings.

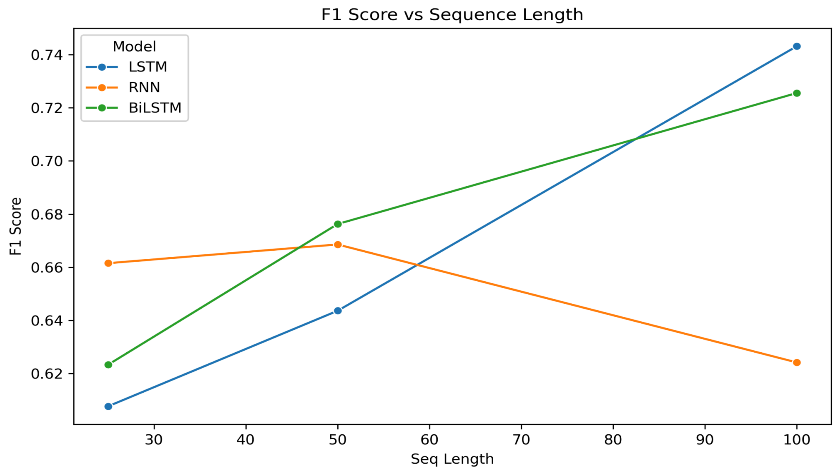
Experiments : Highlighted in Red is best model performance and worst model performance, 50 experiments have been conducted here

Model	**Activation**	**Optimizer**	**Seq Length**	**Grad Clipping**	**Accuracy**	**F1 Score**	**Epoch Time (s)**
LSTM	relu	Adam	50	FALSE	0.73688	0.7349268067	95.8
LSTM	relu	Adam	25	FALSE	0.64216	0.6420898496	56.65
LSTM	relu	Adam	100	FALSE	0.79708	0.796756759	182.64
LSTM	relu	SGD	50	FALSE	0.52196	0.4484843241	84.89
LSTM	relu	RMSprop	50	FALSE	0.72716	0.7263078211	100.15
LSTM	tanh	Adam	50	FALSE	0.71164	0.7116327908	100.26
LSTM	sigmoid	Adam	50	FALSE	0.74792	0.7471155848	107.77
LSTM	relu	Adam	50	TRUE	0.74452	0.7443037714	98.46
LSTM	tanh	Adam	25	FALSE	0.6768	0.6767716821	57.05
LSTM	sigmoid	Adam	100	FALSE	0.80804	0.8075658172	181.41

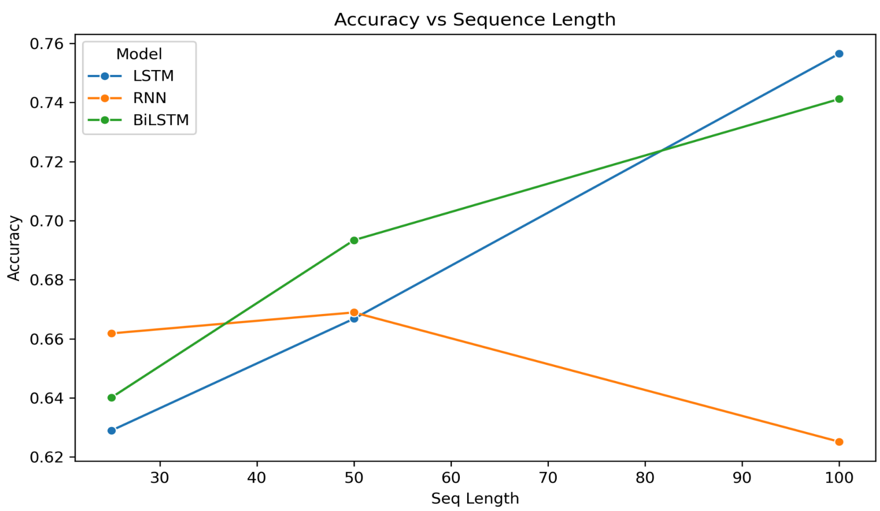
LSTM	relu	RMSprop	25	FALSE	0.67816	0.6771562214	60.16
LSTM	relu	RMSprop	100	FALSE	0.80264	0.8025097626	173.43
LSTM	tanh	RMSprop	50	FALSE	0.70408	0.7000663289	101.14
LSTM	relu	SGD	25	FALSE	0.51852	0.4344468388	46.88
LSTM	relu	SGD	100	FALSE	0.51212	0.4340156346	171.63
LSTM	tanh	SGD	50	FALSE	0.51208	0.4222460408	87.79
LSTM	relu	Adam	50	TRUE	0.7402	0.739915393	99.72
LSTM	relu	Adam	100	TRUE	0.80532	0.8050343054	183.17
LSTM	tanh	RMSprop	100	TRUE	0.81388	0.8132138312	204.6
LSTM	relu	SGD	50	TRUE	0.52152	0.4609206459	88.12
RNN	relu	Adam	50	FALSE	0.6968	0.6966776985	57.61
RNN	relu	Adam	25	FALSE	0.6714	0.6708025857	51.29
RNN	relu	Adam	100	FALSE	0.61236	0.6120479742	97.19
RNN	tanh	Adam	50	FALSE	0.62948	0.6292811517	53.58
RNN	sigmoid	Adam	50	FALSE	0.64476	0.6446433242	60.4
RNN	relu	SGD	50	FALSE	0.63236	0.631383095	40.61
RNN	relu	RMSprop	50	FALSE	0.70648	0.7064670583	48.67
RNN	tanh	RMSprop	50	FALSE	0.6774	0.6763484561	48.92
RNN	relu	Adam	50	TRUE	0.69512	0.6950712817	52.28
RNN	tanh	Adam	25	FALSE	0.65224	0.6522035309	33.55

RNN	sigmoid	Adam	100	FALSE	0.63792	0.6362260146	90.9
BiLSTM	relu	Adam	50	FALSE	0.73944	0.7394397982	150.69
BiLSTM	tanh	Adam	50	FALSE	0.7356	0.7353743696	157.66
BiLSTM	sigmoid	Adam	50	FALSE	0.74408	0.7437004947	156.4
BiLSTM	relu	RMSprop	50	FALSE	0.709	0.7084386118	158.33
BiLSTM	relu	SGD	50	FALSE	0.53116	0.4620083932	141.97
BiLSTM	tanh	RMSprop	50	FALSE	0.72352	0.7235134156	153.59
BiLSTM	tanh	Adam	25	FALSE	0.68188	0.6817521912	90.21
BiLSTM	tanh	Adam	100	FALSE	0.81184	0.811753468	288.52
BiLSTM	relu	Adam	100	FALSE	0.79936	0.7992860088	304.52
BiLSTM	relu	Adam	25	FALSE	0.67592	0.6755878019	83.95
BiLSTM	relu	Adam	50	TRUE	0.7478	0.7476402498	147.13
BiLSTM	tanh	Adam	50	TRUE	0.73968	0.7396670975	174.29
BiLSTM	sigmoid	Adam	50	TRUE	0.74436	0.744143636	173.12
BiLSTM	relu	RMSprop	25	FALSE	0.66544	0.6630671279	87.5
BiLSTM	relu	RMSprop	100	FALSE	0.78972	0.788125155	292.18
BiLSTM	tanh	RMSprop	100	TRUE	0.78964	0.78913371	298.45
BiLSTM	relu	SGD	25	FALSE	0.53708	0.4726408257	67.83
BiLSTM	relu	SGD	100	FALSE	0.51524	0.4395568941	259.87
BiLSTM	tanh	SGD	50	TRUE	0.51912	0.4185322087	131.39

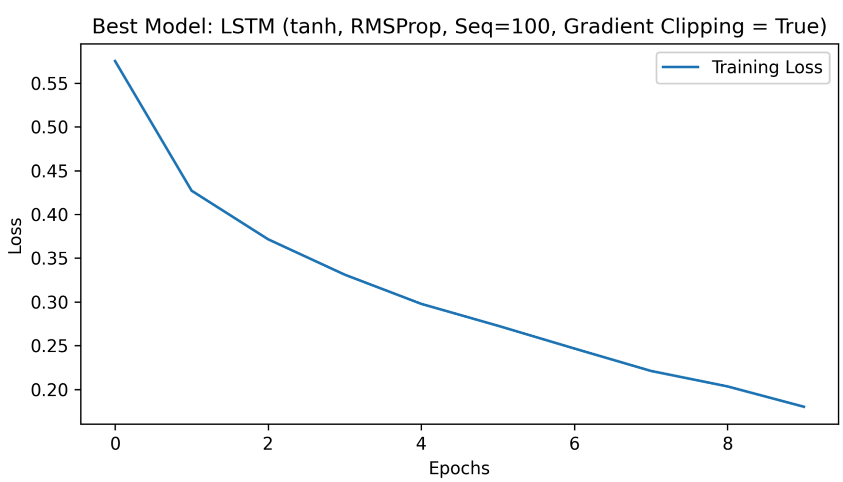
<u>Plots:</u>



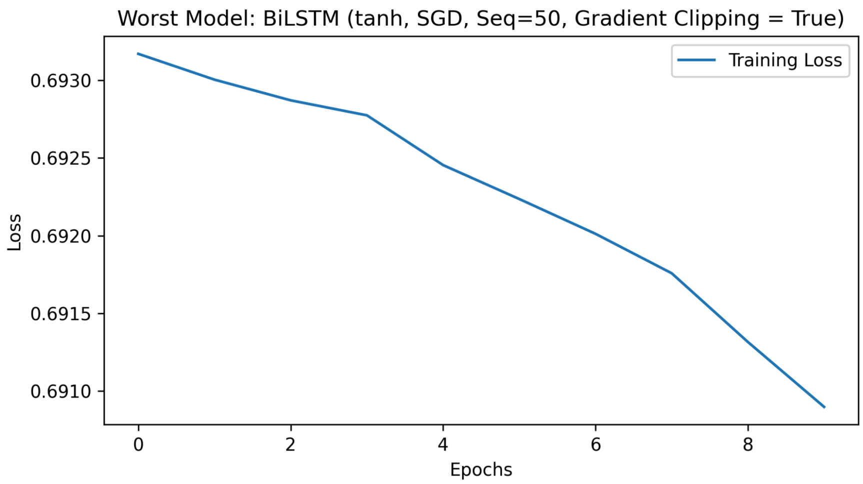
F1 Score vs Sequence Length
Model
LSTM
RNN
BiLSTM
F1 Score
0.74
0.72
0.70
0.68
0.66
0.64
0.62
30
40
50
60
70
80
90
100
Seq Length



Accuracy vs Sequence Length
Model
LSTM
RNN
BiLSTM
Accuracy
0.76
0.74
0.72
0.70
0.68
0.66
0.64
0.62
30
40
50
60
70
80
90
100
Seq Length



Best Model: LSTM (tanh, RMSProp, Seq=100, Gradient Clipping = True)
Training Loss
Loss
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0
2
4
6
8
Epochs



The plots for Accuracy and F1 score vs the sequence length shows the relationship between the increase in context window and performance. LSTM's consistently improved with longer sequence lengths, with the F1 score and accuracy showing an improvement. The LSTM architecture therefore improves with more contextual information, capturing different sentiment patterns. BiLSTM also benefited from longer sequences but it's improvement plateaued earlier in comparison to LSTM's and this behavior aligns with its bidirectional behavior as it processes sequences from both ends. RNN's, however, degraded as sequence lengths increased. It's accuracy and F1 score dropped with the increase in sequence length, which relates to the vanishing gradient problem, where recurrence units are unable to maintain gradients over longer sequences

<u>Discussion:</u>

Which configuration performed best?

The configuration which performed best was an LSTM model using a tanh activation function, RMSProp for optimization and gradient clipping being true. It had an accuracy of around 0.81388 and an F1 score of 0.81321. The reasons why this configuration performed best was : -

- LSTM models long term dependencies and prevents vanishing gradients
- Tanh as an activation function balances positive and negative gradient flow, leading to smoother convergence
- RMSProp adapts learning rate per parameter, which complements LSTM's internal mechanism

- Gradient clipping helped with updates for 100 token sequences, avoiding exploding gradients

However, the training time for this model was longer (204.6 seconds per epoch), this model provided the most accurate results

How did sequence length or optimizer affect performance?

Sequence length affects performance in the following manner as:

- Increasing sequence length from 25 to 500 and then 100 tokens showed a consistent improvement in F1 score and accuracy
- The shorter sequences often lost context due to the issue of long term dependencies, not performing as well as longer sequences
- The longer sequences performed better compared to the shorter sequences but the training time also increased proportionally.
- The 100 token length was found to be the most optimal in terms of performance

The optimizer affected performance in the following manner by :-

- RMSProp had the best performance for these models. It performed best when paired with LSTM's and maintained a steady learning rate
- Adam also performed well, especially with ReLU activations, but also tended to plateau early
- SGD was least effective and often converging slowly, leading to low F1 scores due to the variance in gradient

How did gradient clipping impact stability?

Gradient clipping significantly improved training stability and convergence for longer sequences. Without clipping, a lot of the models started showing diverging loss at later epochs. Enabling gradient clipping helped in preventing exploding gradients during backpropagation and produced a smoother loss curve. It also resulted in a higher F1 score across runs. Therefore, it was useful for models using longer sequences and increased depth.

Conclusion

After conducting 50 experiments over various runs and settings, the optimal configuration was LSTM with a Tanh activation, RMSProp optimizer, Sequence length of 100 and gradient clipping set as True.

This configuration offered the best accuracy and F1 score among the 50 experiments. The LSTM model effectively captured long distance dependencies in the textual data, while the Tanh function provided smoother convergence by maintaining balanced gradients. The RMSProp optimizer helped with training stability by adjusting the learning rate per parameter and gradient clipping helped mitigate exploding gradients during the process of backpropagation.

However, the tradeoff was that the model took a longer time to train per epoch (204.6 s per epoch). Despite this, it consistently performed well compared to other configurations which took lesser time to train. This suggests that this model setup offers the best balance between performance and efficiency on a CPU system and is very reliable for classification on the IMDB movie reviews dataset.