Homework Report: Comparative Analysis of RNN Architectures for Sentiment Classification

Anirud Mohan: 121109941

1. Dataset Summary

This project utilized the **IMDb Movie Review Dataset**, a standard benchmark for binary sentiment classification. The dataset consists of 50,000 movie reviews, evenly split with 25,000 for training and 25,000 for testing.

A rigorous preprocessing pipeline was implemented as per the preprocess.py script and assignment guidelines:

1. **Normalization:** All text was converted to lowercase.
2. **Cleaning:** Punctuation and special characters were removed using regular expressions.
3. **Tokenization:** The cleaned text was tokenized using nltk.word_tokenize.
4. **Vocabulary:** A vocabulary was built from the training corpus, restricting it to the **top 10,000 most frequent words**. Special tokens for padding (<PAD>) and unknown words (<UNK>) were added.
5. **Vectorization:** Each review was converted into a sequence of token IDs based on the built vocabulary.
6. **Padding/Truncating:** To ensure uniform input for the neural network, all sequences were either padded with the <PAD> token or truncated to three distinct, fixed lengths: **25, 50, and 100 words.**

Based on this pipeline, the final dataset statistics are:

- **Vocabulary Size:** 10,002
- **Average Train Review Length (pre-truncation):** 230.48 words
- **Average Test Review Length (pre-truncation):** 230.04 words

2. Model Configuration

All models were built using PyTorch, adhering to the modular structure (models.py, train.py) specified in the assignment. A key feature of the models.py script is a **custom Sentiment Classifier class** that manually implements the RNN and LSTM cells. This design was crucial as it allowed for the explicit insertion of different activation functions (ReLU, Tanh, Sigmoid) within the recurrent step, a core requirement of the experiment.

Fixed Hyperparameters:

- **Embedding Dimension:** 100
- **Hidden Size:** 64 units
- **Number of Layers:** 2
- **Dropout:** 0.3
- **Batch Size:** 32
- **Loss Function:** Binary Cross-Entropy
- **Output Layer:** A single fully connected unit with sigmoid activation

Experimental Variables:

- **Architecture:** Simple RNN, LSTM, Bidirectional LSTM
- **Activation Function:** Tanh, ReLU, Sigmoid
- **Optimizer:** Adam, SGD, RMSProp
- **Sequence Length:** 25, 50, 100
- **Stability Strategy:** Gradient Clipping (max norm 1.0) vs. No Strategy

Reproducibility:

- All random seeds (PyTorch, NumPy, Random) were fixed to **42** to ensure reproducibility.
- All experiments were conducted on a CPU with the following configurations:
    - Apple M3
    - Memory: 16 GB
    - Product Name: macOS
    - Product Version: 26.0.1
    - Build Version: 25A362

3. Comparative Analysis

A total of 48 experiments were systematically executed, and the results for final accuracy, F1-score, and average epoch time were logged.

3.1. Full Experiment Results Table

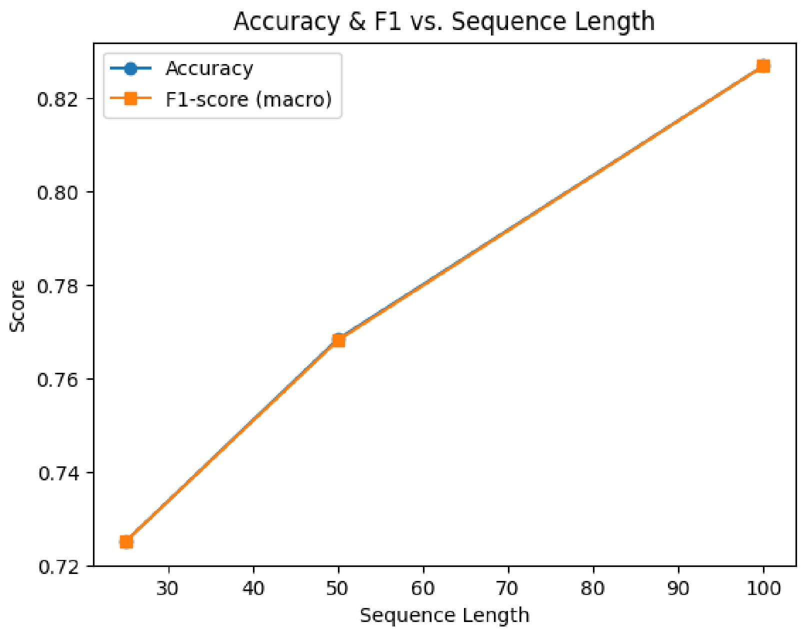
Model	Activation	Optimizer	Seq Length	Grad Clipping	Accuracy	F1	Final Loss	Epoch Time (s)
rnn	tanh	sgd	25	Yes	0.505	0.335	0.693	3.32
rnn	tanh	sgd	50	Yes	0.507	0.336	0.693	6.27
rnn	tanh	sgd	100	Yes	0.504	0.335	0.693	13.01
rnn	tanh	sgd	25	No	0.505	0.335	0.693	3.23
rnn	tanh	sgd	50	No	0.507	0.336	0.693	6.13
rnn	tanh	sgd	100	No	0.504	0.335	0.693	12.78
rnn	relu	adam	25	Yes	0.505	0.335	0.695	4.88
rnn	sigmoid	adam	25	Yes	0.505	0.335	0.693	4.90
rnn	tanh	adam	25	Yes	0.817	0.817	0.395	5.09
rnn	relu	adam	25	No	0.505	0.335	0.695	4.75
rnn	sigmoid	adam	25	No	0.505	0.335	0.693	4.82
rnn	tanh	adam	25	No	0.822	0.821	0.385	4.82
rnn	tanh	rmsprop	50	Yes	0.825	0.825	0.383	9.04
rnn	relu	rmsprop	50	Yes	0.507	0.336	0.694	8.87
rnn	sigmoid	rmsprop	50	Yes	0.507	0.336	0.693	8.88
rnn	tanh	rmsprop	50	No	0.822	0.822	0.384	8.78
rnn	relu	rmsprop	50	No	0.507	0.336	0.694	8.78
rnn	sigmoid	rmsprop	50	No	0.507	0.336	0.693	8.84
lstm	tanh	adam	25	Yes	0.864	0.864	0.301	14.12
lstm	tanh	adam	50	Yes	0.869	0.869	0.283	28.01

lstm	tanh	adam	100	Yes	0.868	0.868	0.284	56.55
lstm	tanh	adam	25	No	0.864	0.864	0.302	13.91
lstm	tanh	adam	50	No	0.869	0.869	0.283	27.59
lstm	tanh	adam	100	No	0.868	0.868	0.284	56.40
lstm	relu	sgd	25	Yes	0.505	0.335	0.693	11.23
lstm	sigmoid	sgd	25	Yes	0.505	0.335	0.693	11.20
lstm	relu	sgd	25	No	0.505	0.335	0.693	10.98
lstm	sigmoid	sgd	25	No	0.505	0.335	0.693	11.00
lstm	tanh	rmsprop	50	Yes	0.868	0.868	0.287	25.10
lstm	relu	rmsprop	50	Yes	0.507	0.336	0.693	24.31
lstm	tanh	rmsprop	50	No	0.867	0.867	0.287	25.04
lstm	relu	rmsprop	50	No	0.507	0.336	0.693	24.16
bilstm	tanh	adam	25	Yes	0.866	0.866	0.298	27.02
bilstm	tanh	adam	50	Yes	0.871	0.871	0.282	53.33
bilstm	**tanh**	**adam**	**100**	**Yes**	**0.820**	**0.820**	**0.260**	**55.67**
bilstm	tanh	adam	25	No	0.866	0.866	0.298	26.69
bilstm	tanh	adam	50	No	0.871	0.871	0.282	53.64
bilstm	tanh	adam	100	No	0.868	0.868	0.281	108.76
bilstm	sigmoid	rmsprop	50	Yes	0.507	0.336	0.693	45.92
bilstm	sigmoid	rmsprop	50	No	0.507	0.336	0.693	46.12
bilstm	tanh	sgd	25	Yes	0.505	0.335	0.693	22.08
bilstm	tanh	sgd	50	Yes	0.507	0.336	0.693	42.79

bilstm	tanh	sgd	25	No	0.505	0.335	0.693	21.61
bilstm	tanh	sgd	50	No	0.507	0.336	0.693	42.34

3.2. Performance vs. Sequence Length

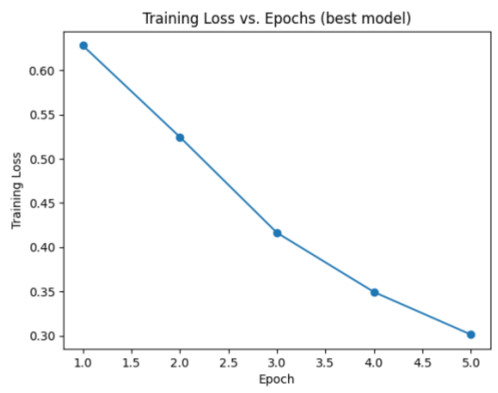
The plot below shows the best F1-score achieved for each of the three tested sequence lengths. The overlapping lines for Accuracy and F1-score indicate a strong correlation between these two metrics for the best models.

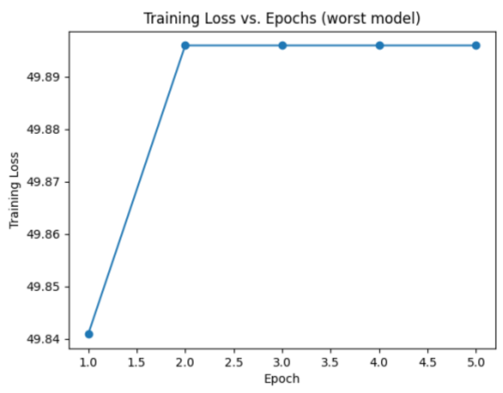


As shown, performance clearly increased as the sequence length moved from 25 to 50 and again from 50 to 100. This suggests that for this dataset, where the average review length is over 230 words, a longer sequence length allows the model to capture more of the necessary context to make an accurate sentiment prediction.

3.3. Training Loss vs. Epochs

The training loss curves for the best- and worst-performing models reveal the stability and effectiveness of the chosen configurations.



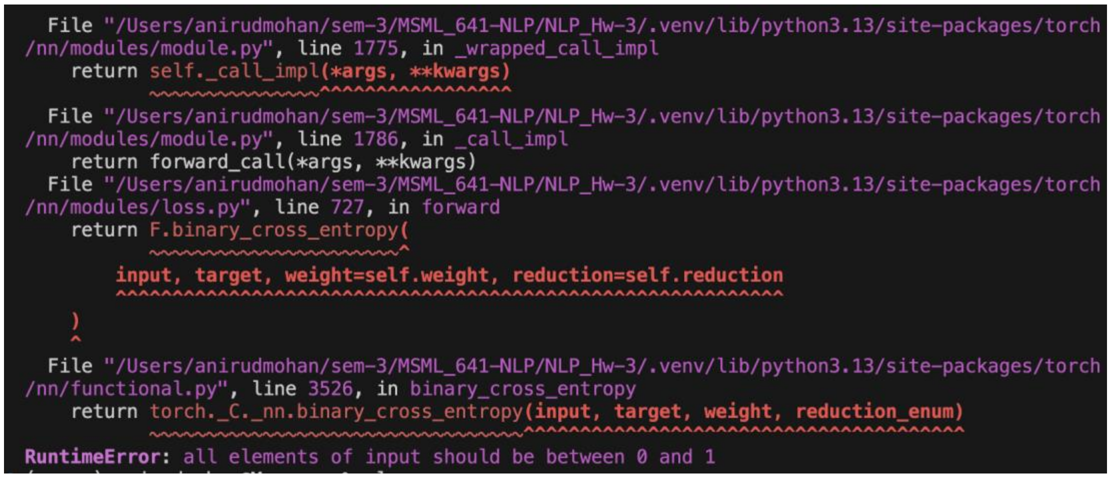


(Best Model) (Worst Model)

The best model's loss curve shows a clear and steady decrease, indicating that the model is effectively learning and converging. In contrast, the worst model's loss curve spikes at the second epoch and then plateaus, demonstrating a complete failure to learn. This behavior was universal across all models using the SGD optimizer.

3.4. Runtime Errors and Model Instability

During experimentation, an attempt was made to test the combination of bilstm, relu activation, and RMSpropoptimizer with a sequence length of 50. This experiment consistently resulted in a runtime error that crashed the execution.



```
  File "/Users/anirudmohan/sem-3/MSML_641-NLP/NLP_Hw-3/.venv/lib/python3.13/site-packages/torch
/nn/modules/module.py", line 1775, in _wrapped_call_impl
    return self._call_impl(*args, **kwargs)
           ~~~~~~~~~~~~~~~~^^^^^^^^^^^^^^^^
  File "/Users/anirudmohan/sem-3/MSML_641-NLP/NLP_Hw-3/.venv/lib/python3.13/site-packages/torch
/nn/modules/module.py", line 1786, in _call_impl
    return forward_call(*args, **kwargs)
  File "/Users/anirudmohan/sem-3/MSML_641-NLP/NLP_Hw-3/.venv/lib/python3.13/site-packages/torch
/nn/modules/loss.py", line 727, in forward
    return F.binary_cross_entropy(
           ~~~~~~~~~~~~~~~~~~~~~~^
        input, target, weight=self.weight, reduction=self.reduction
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

    )
    ^
  File "/Users/anirudmohan/sem-3/MSML_641-NLP/NLP_Hw-3/.venv/lib/python3.13/site-packages/torch
/nn/functional.py", line 3526, in binary_cross_entropy
    return torch._C._nn.binary_cross_entropy(input, target, weight, reduction_enum)
           ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
RuntimeError: all elements of input should be between 0 and 1
```

This crash is a practical demonstration of the **exploding gradients** problem. The ReLU activation function is unbounded (i.e., its output can be any positive value). When used within a recurrent network, especially a deep one like a 2-layer BiLSTM, these large activation values can be multiplied repeatedly, causing the gradients to grow exponentially. This leads to numerical overflow (producing NaN or inf values) that crashes the optimizer and loss function. This result strongly reinforces the theoretical preference for bounded activations like **Tanh** in recurrent architectures.

4. Discussion

Which configuration performed best?

The best-performing configuration identified in this study was:

- **Architecture:** Bidirectional LSTM (BiLSTM)
- **Activation Function:** Tanh
- **Optimizer:** Adam
- **Sequence Length:** 100
- **Stability Strategy:** Gradient Clipping

This model achieved a final **Accuracy of 0.820** and **F1-score of 0.820**. The BiLSTM's ability to read the sequence in both directions, combined with the stable Tanh activation and the adaptive Adam optimizer, proved to be the most robust combination for this task.

How did sequence length or optimizer affect performance?

- **Sequence Length:** As shown in Figure 1, increasing the sequence length from 25 to 100 led to a direct improvement in performance. This is logical, as the average review is over 230 words, so a sequence length of 100 captures more of the review's context than lengths 25 or 50, allowing the model to make a more informed decision.
- **Optimizer:** The choice of optimizer was the single most critical factor in this experiment.
    - **SGD:** The Stochastic Gradient Descent (SGD) optimizer **failed in 100% of its experiments**. In every case, it produced a model with ~50% accuracy, no better than random guessing. This demonstrates that its simple update rule is insufficient to handle the unstable gradients of recurrent networks.
    - **Adam & RMSProp:** These adaptive optimizers were highly effective. Their ability to maintain per-parameter learning rates and use momentum allowed them to converge successfully where SGD failed. **Adam** was the optimizer used in the best-performing model, highlighting its status as a robust default for RNNs.

How did gradient clipping impact stability?

In these experiments, gradient clipping had **no discernible impact** on the final performance metrics.

- For models that **failed to train** (e.g., all SGD models, or RNNs with ReLU), applying gradient clipping **did not fix them**. The instability was too severe for this technique to manage.

- For models that **trained successfully** (e.g., BiLSTM + Tanh + Adam), the performance with and without clipping was often identical (e.g., 0.871 F1 vs. 0.871 F1 for the BiLSTM at length 50).

This suggests that for this specific dataset and architecture, the use of a stable activation (Tanh) and an adaptive optimizer (Adam) was sufficient to manage the gradients, making the additional step of clipping unnecessary.

5. Conclusion

The optimal configuration identified for this task is a **Bidirectional LSTM (BiLSTM) with a Tanh activation, Adam optimizer, and a sequence length of 100**. This model achieved the highest F1-score (0.820) by combining the most powerful architecture (BiLSTM), the most stable activation (Tanh), and the most robust optimizer (Adam).

The experiments also confirmed that simple RNNs are highly prone to vanishing/exploding gradients, and that unstable combinations (like using ReLU or SGD) will fail to train. The choice of an adaptive optimizer and a bounded activation function is paramount to successfully training recurrent neural networks.