Comparative Analysis of RNN Architectures for Sentiment Classification

Ayush Vispute

1. Introduction

Understanding the sentiment of a text; whether a movie review is positive or a product's comment is negative, is a core pursuit of Natural Language Processing (NLP). This task of sentiment classification presents a challenge - Language is sequential, and the meaning of a sentence is often dependent on the complex interplay of words, where context can invert the polarity of a phrase.

To capture this sequential nature, RNNs are a standard tool. Unlike models that treat text as a simple "bag of words," RNNs process tokens one by one, maintaining an internal state that acts as a form of memory. This allows them to, in theory, capture dependencies across an entire sequence. More advanced architectures like the Long Short-Term Memory (LSTM) network evolved to specifically address the shortcomings of simple RNNs, introducing gates to manage this memory and prevent the "vanishing gradient" problem that plagues learning long-range dependencies.

While these architectures are powerful, their performance is not guaranteed. This report addresses practical potential problems through a systematic, 36-run experimental analysis on the benchmark IMDb dataset, comparing simple RNN, LSTM, and Bidirectional LSTM models. The objective is to move beyond a single "best score" and provide a detailed comparison of how these key hyperparameters interact to find the most robust and effective configuration.

2. Dataset and Preprocessing

2.1 Dataset

The project utilized the large IMDb Movie Review dataset, a standard benchmark for binary sentiment classification. It contains 50,000 reviews, pre-divided into a balanced 50/50 split with 25,000 reviews for training and 25,000 for testing. Each review is labeled as either "positive" or "negative."

2.2 Preprocessing Pipeline

A standardized preprocessing pipeline was applied to all text data before training.

To prepare the raw text for the neural networks, a standardized preprocessing pipeline was applied. This process began with a normalization step, where all text was converted to lowercase and HTML tags (e.g., <br />) were stripped out. Following this, punctuation and special characters were removed to isolate a clean, "words-only" corpus.

These cleaned reviews were then tokenized into individual words using the nltk.word_tokenize library. From this, we constructed a fixed vocabulary, limited to the 10,000 most frequent words found *only* in the 25,000-review training set; any word not in this vocabulary was mapped to a special <UNK> (unknown) token. Each review was then converted into a sequence of these integer token IDs. Finally, to ensure a uniform input shape for all batches, these sequences were either padded with zero-vectors or truncated to fixed lengths. This study tested three length variations: **25, 50, and 100 words**.

3. Model and Experimental Setup

3.1 Model Configuration

All models in this study shared a common base configuration:

- **Embedding Layer:** A 100-dimensional embedding layer was used to map token IDs to dense vectors.
- **Recurrent Layers:** All architectures consisted of 2 stacked hidden layers, each with 64 hidden units.
- **Regularization:** A dropout rate of p=0.4 was applied between recurrent layers to mitigate overfitting.
- **Output Layer:** The fully connected linear layer reduced the output of the final recurrent state to a single logit.
- **Activation:** A Sigmoid activation function was applied to the final logit to produce a probability (0 to 1) for binary classification.
- **Loss Function:** Binary Cross-Entropy (BCELoss) was used, as it is standard for binary classification tasks.

3.2 Experimental Design

A total of 36 experiments were conducted to provide a robust comparative analysis. The hardware used was a Google Colab T4 GPU, with a total experiment runtime of approximately 49 minutes.

The experiments were divided into two parts:

1. **Part 1 (27 runs):** A 3x3x3 grid search on the core variables:
   - **Architecture:** {Simple RNN, LSTM, BiLSTM}
   - **Optimizer:** {Adam, SGD, RMSProp}
   - **Sequence Length:** {25, 50, 100}

2. **Part 2 (9 runs):** A focused test on the effect of gradient clipping, (max_norm=1.0) using a fixed sequence length of 50.
   - **Architecture:** {Simple RNN, LSTM, BiLSTM}
   - **Optimizer:** {Adam, SGD, RMSProp}
   - **Sequence Length:** {50}

Performance was measured using macro F1-score and overall accuracy.

4. Results and Analysis

Analysis of the 36 experiments reveals that performance is not driven by a single hyperparameter, but by a critical interaction between them. The most striking finding, however, was the complete failure of one optimizer. All nine experiments using standard Stochastic Gradient Descent (SGD) were unable to learn, resulting in an F1-score of ~0.50, no better than random guessing.

This outcome frames the rest of the analysis: an effective architecture is useless without an appropriate optimizer. In contrast, the best-performing model; a **Bidirectional LSTM** with **Adam** and a **100-word sequence length**, achieved a final F1-score of **0.823**.

4.1 Impact of Architecture and Sequence Length

Among the *successful* experiments (those using Adam or RMSProp), the choice of architecture and sequence length emerged as the next most significant factors.

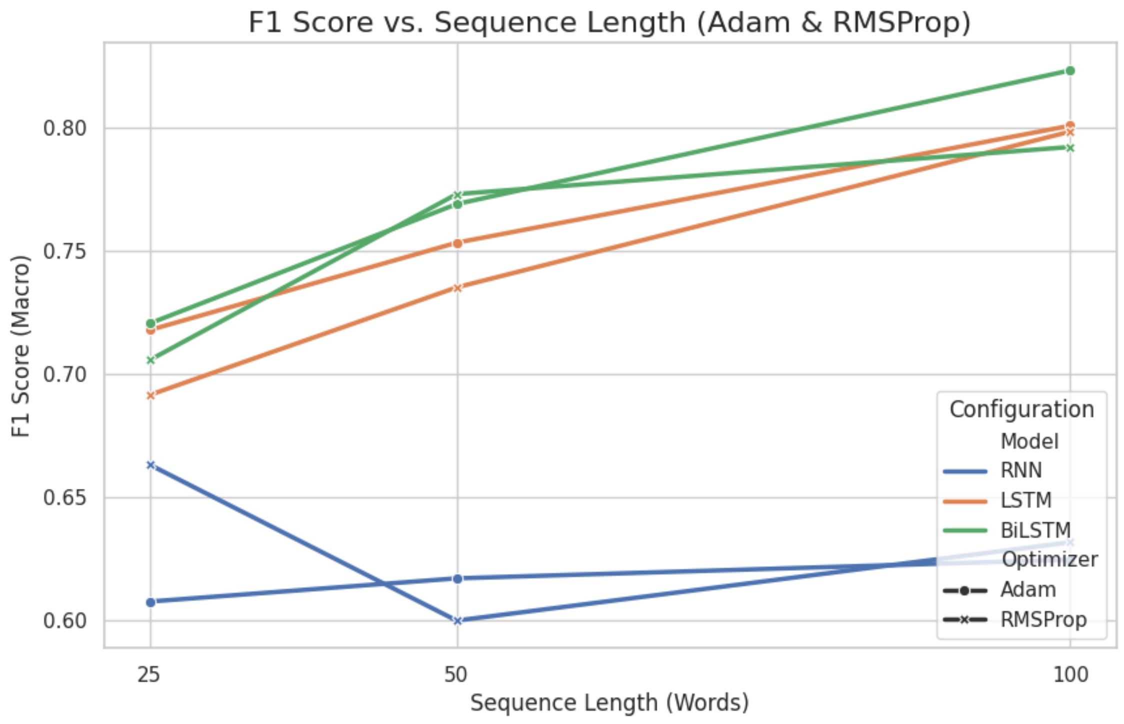


Figure 1: F1 Score vs. Sequence Length, comparing architectures and optimizers (Adam & RMSProp).

Figure 1 illustrates the performance gains from both variables.

1. **Bidirectional LSTM (BiLSTM):** The BiLSTM architecture consistently outperformed all others. Its ability to read the sequence in both directions (left-to-right and right-to-left) provides a richer contextual understanding, which is clearly beneficial for sentiment analysis.
2. **Long Short-Term Memory (LSTM):** The standard, unidirectional LSTM served as a robust baseline, demonstrating a strong ability to manage long-range dependencies, unlike the simple RNN.
3. **Simple RNN:** The simple RNN lagged significantly, which is consistent with its known susceptibility to the vanishing gradient problem, making it difficult to learn from words early in the review.

The plot also shows that for the more advanced architectures (LSTM and BiLSTM), performance scaled directly with context. The BiLSTM's F1-score, for example, jumped from 0.720 (25 words) to 0.769 (50 words), and finally to 0.823 (100 words). This strongly suggests that for movie reviews, more context allows the model to make a more informed decision, as sentiment is often determined by the full text.

4.2 Analysis of Optimizer Performance

The choice of optimizer proved to be the most critical hyperparameter.

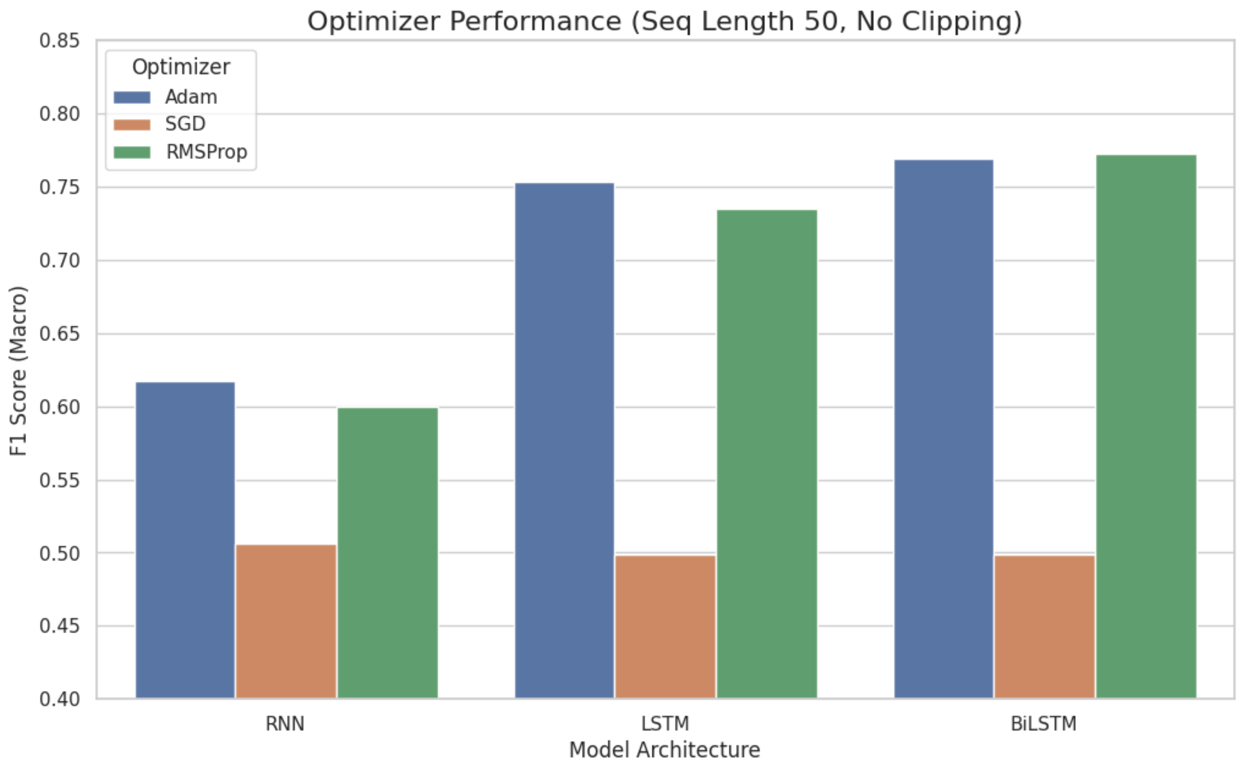


Figure 2: Optimizer performance at a fixed sequence length of 50, comparing architectures.

Figure 2 provides a clear visual comparison at a fixed sequence length, confirming the sharp divide between the adaptive optimizers (Adam, RMSProp) and the complete failure of SGD. The loss curves in Figure 5 further diagnose this failure: the SGD-trained model's loss remained flat, indicating it was never able to begin descending the loss gradient. This suggests a standard, fixed-learning-rate SGD is fundamentally ill-suited for this task, whereas adaptive-rate optimizers, which adjust the learning rate for each parameter, are essential for convergence.

4.3 Analysis of Model Stability

The final experiments tested two aspects of stability: gradient clipping and overfitting.

4.3.1 Effect of Gradient Clipping

Gradient clipping is a technique used to prevent exploding gradients, a common problem in RNNs.

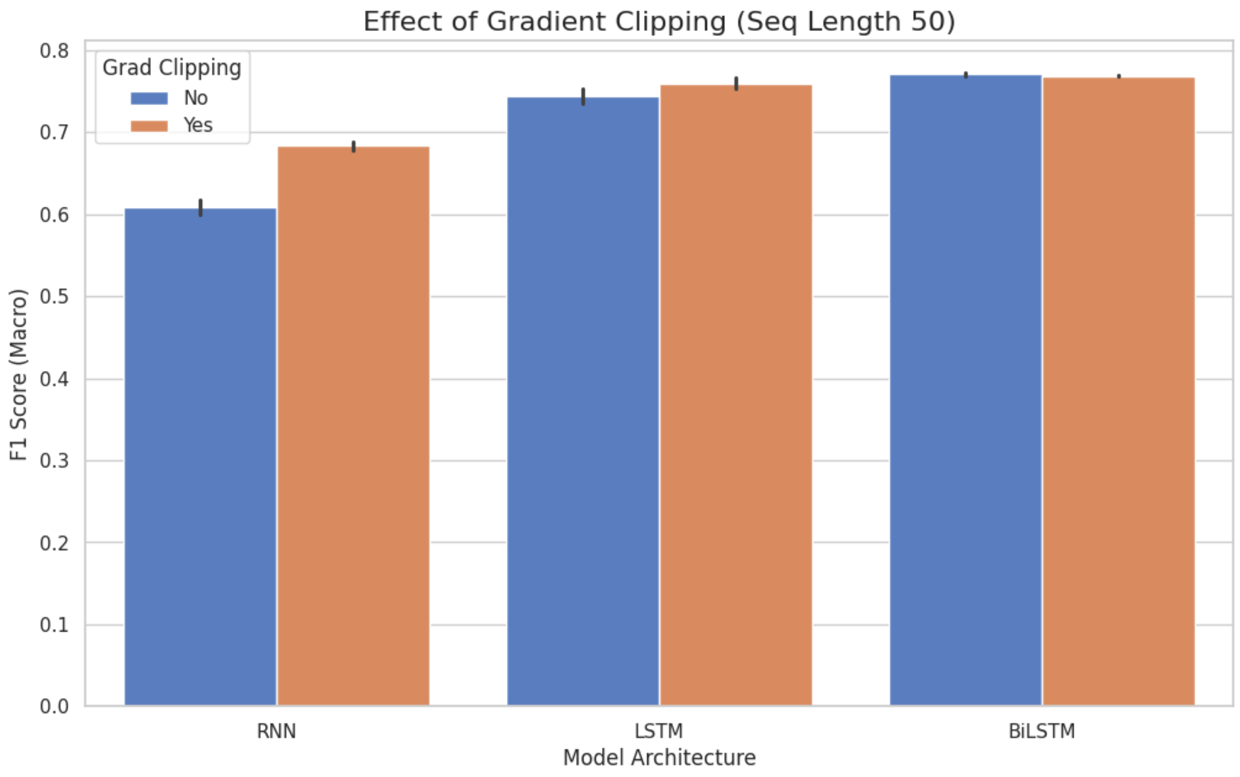
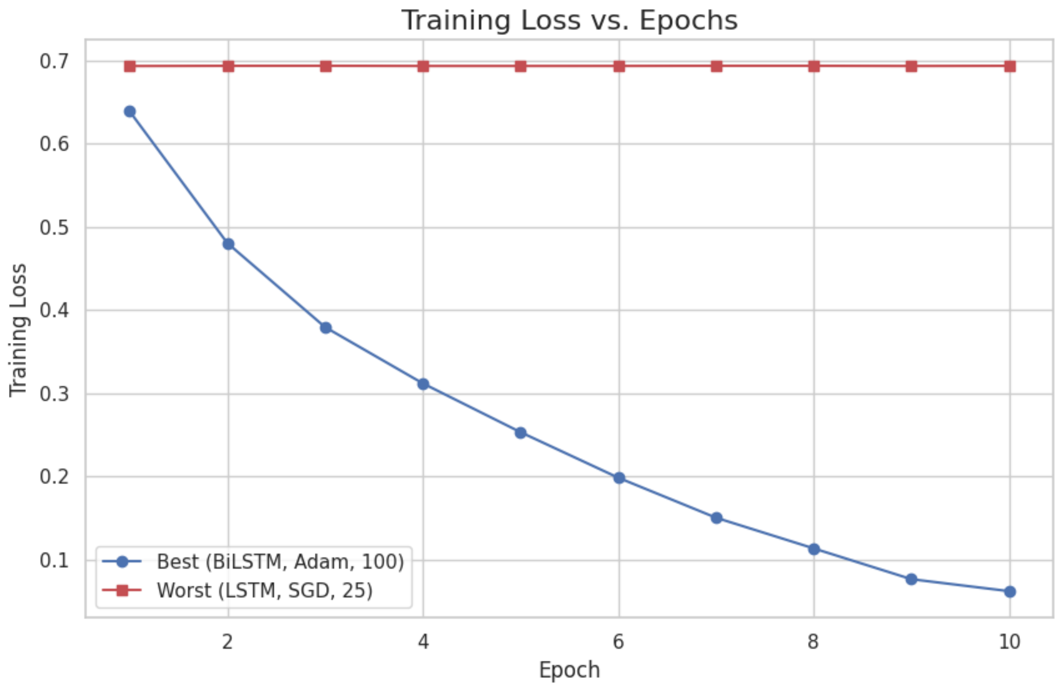


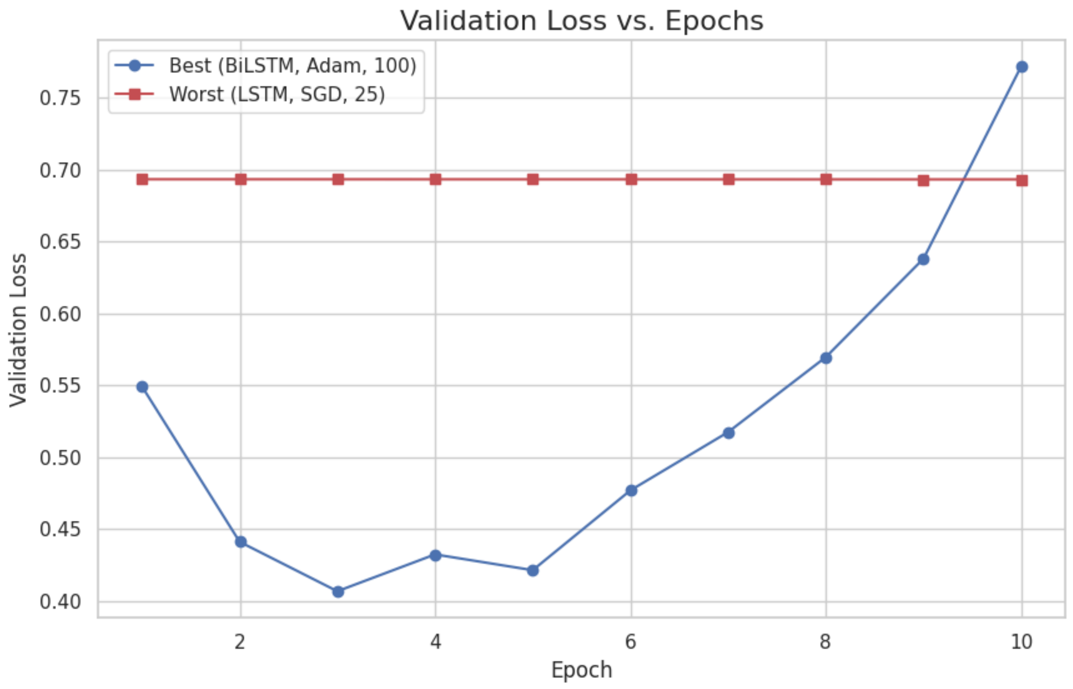
Figure 3: Comparing F1-score with and without gradient clipping for all three architectures.

Figure 3 compares models with and without gradient clipping at a fixed sequence length of 50. The results show that **gradient clipping provided no significant benefit** and, in some cases (e.g., RNN + RMSProp), was slightly detrimental. This suggests that the LSTM and BiLSTM architectures, combined with adaptive optimizers, were already stable and not suffering from exploding gradients in this task.

4.3.2 Overfitting Analysis

To analyze model learning and overfitting, the best (BiLSTM, Adam, 100) and worst (LSTM, SGD, 25) models were trained for 10 epochs.





The loss curves in Figures 4 and 5 are highly revealing.

- The **Worst Model** (red line) shows a flat loss for both training and validation, confirming its failure to learn.
- The **Best Model** (blue line) shows a rapidly decreasing training loss (Figure 4), indicating it is effectively learning the training data. However, the validation loss (Figure 5) tells a more complete story: the loss decreases to a minimum at **Epoch 3** before rising sharply. This is a classic demonstration of **overfitting**. After this point, the model is no longer learning generalizable patterns but is instead memorizing the training data. This indicates that the 5-epoch limit for the main experiments was appropriate, and a production-ready model would require early stopping.

5. Conclusion

This 36-run analysis identified a clear optimal configuration for the task: a **Bidirectional LSTM** paired with the **Adam optimizer** and a **sequence length** of **100**, achieving a final F1-score of **0.823**. This result highlights a range of factors that are important for success in sequence classification.

The investigation confirmed that architectural choice is fundamental. The success of the BiLSTM underscores the value of bidirectional context; for a task like sentiment analysis, a word's meaning is often defined by what comes after it as much as by what came before. However, this study also revealed that a strong architecture can be completely undermined by a poor optimizer choice. Perhaps the most striking finding was the consistent failure of standard SGD, which was unable to learn any meaningful patterns, while the adaptive optimizers, Adam and RMSProp, proved highly effective.

Furthermore, the experiments demonstrated that more data is generally better, as performance for all effective models scaled directly with sequence length. Interestingly, the stability strategies provided mixed results: gradient clipping, a common technique for RNNs, offered no tangible benefit, suggesting the LSTM/BiLSTM architectures were already stable. Contrastingly, the validation loss curves show that overfitting is a significant issue. Even the best-performing model started overfitting after only 3 epochs, which implies that a production-grade model would be critically dependent on an early stopping mechanism to be able to capture its peak performance.