Homework 3 Report: Comparative Analysis of RNN Architectures

https://github.com/madhurlak0810/homework3_sentiment_classification

Madhur Lakshmanan
UID:121324702

Contents

Abstract

This report presents a comprehensive comparative analysis of Recurrent Neural Network (RNN) architectures for sentiment classification on the IMDb movie reviews dataset. We systematically evaluated 162 different configurations across RNN, LSTM, and Bidirectional LSTM models, varying activation functions, optimizers, sequence lengths, and gradient clipping strategies. Our experiments achieved a maximum accuracy of 82.43% using a Bidirectional LSTM with sigmoid activation and RMSprop optimizer on sequences of length 100. Key findings indicate that sequence length has the most significant impact on performance, while optimizer choice substantially affects convergence and final accuracy. The results demonstrate that modern RNN architectures can achieve competitive performance on sentiment classification tasks with proper hyperparameter tuning.

Keywords: Sentiment Analysis, RNN, LSTM, Natural Language Processing, Deep Learning

1 Introduction

1.1 Problem Statement

Sentiment classification is a fundamental task in Natural Language Processing that involves automatically determining the emotional tone or opinion expressed in text. This project focuses on binary sentiment classification of movie reviews, categorizing them as positive or negative using various RNN architectures.

1.2 Objectives

- Implement and compare RNN, LSTM, and Bidirectional LSTM architectures
- Analyze the impact of different activation functions (ReLU, Sigmoid, Tanh)
- Evaluate various optimizers (Adam, SGD, RMSprop) on training dynamics
- Investigate the effect of sequence length on model performance
- Assess the impact of gradient clipping on training stability
- Provide comprehensive statistical analysis of hyperparameter effects

1.3 Dataset Overview

- **Dataset**: IMDb Movie Review Dataset
- **Size**: 50,000 reviews (25,000 training, 25,000 testing)
- **Task**: Binary classification (positive/negative)
- **Preprocessing**: Top 10,000 vocabulary, sequence lengths: 25, 50, 100
- **Class Distribution**: Balanced dataset with equal positive/negative samples

2 Methodology

2.1 Data Preprocessing

2.1.1 Text Cleaning

- Conversion to lowercase for consistency
- Removal of HTML tags, URLs, and special characters
- Tokenization using space-based word splitting
- Vocabulary filtering to top 10,000 most frequent words

2.1.2 Vocabulary Construction

- Built from training data only to prevent test set leakage
- Special tokens: `<PAD>` (index 0), `<UNK>` (index 1) for unknown words
- Word-to-index mapping for neural network compatibility

2.1.3 Sequence Processing

- Padding sequences shorter than target length with `<PAD>` tokens
- Truncating sequences longer than target length from the end
- Three sequence length configurations: 25, 50, 100 words

2.2 Model Architectures

2.2.1 Base Configuration

All models share the following configuration:

- **Embedding Layer**: 100 dimensions (trainable)
- **Hidden Layers**: 2 layers with 64 units each
- **Dropout**: 0.3 for regularization
- **Output Layer**: Single linear layer with sigmoid activation
- **Loss Function**: Binary cross-entropy

2.2.2 Architecture Variations

Vanilla RNN (`SentimentRNN`)

- Basic recurrent layer with configurable activation
- Processes sequences sequentially left-to-right
- Prone to vanishing gradient problems on longer sequences

LSTM (`SentimentLSTM`)

- Long Short-Term Memory cells with forget, input, and output gates
- Addresses vanishing gradient problem through gating mechanisms
- Better long-term dependency modeling

Bidirectional LSTM (`SentimentBiLSTM`)

- Processes sequences in both forward and backward directions
- Combines information from past and future context
- Enhanced sequence understanding through bidirectional processing

2.3 Experimental Design

2.3.1 Systematic Parameter Variation

We conducted a full grid search across all hyperparameter combinations:

Category	Variations
Architecture	RNN, LSTM, Bidirectional LSTM
Activation	ReLU, Sigmoid, Tanh
Optimizer	Adam, SGD, RMSprop
Sequence Length	25, 50, 100 words
Gradient Clipping	Enabled vs Disabled

Table 1: Hyperparameter grid search space.

Total Experiments: $3 \times 3 \times 3 \times 3 \times 2 = 162$ configurations

2.3.2 Training Parameters

- **Batch Size**: 32 samples
- **Learning Rate**: 0.001 for all optimizers
- **Epochs**: Maximum 20 with early stopping
- **Early Stopping**: Patience of 5 epochs, minimum delta of 0.001
- **Gradient Clipping**: Maximum norm of 1.0 when enabled
- **Seed**: Fixed at 42 for reproducibility

2.3.3 Evaluation Metrics

- **Primary**: Test accuracy and F1-score (macro-averaged)
- **Secondary**: Training time per epoch for efficiency analysis
- **Additional**: Precision, Recall, ROC AUC for comprehensive evaluation

3 Results

3.1 Overall Performance Summary

Best Performing Models (Top 5):

Rank	Model	Activation	Optimizer	Seq Len	Grad Clip	Acc (%)	F1	Time (s)
1	BiLSTM	Sigmoid	RMSprop	100	No	82.43	82.43	2.43
2	BiLSTM	Sigmoid	Adam	100	No	82.26	82.26	2.39
3	LSTM	ReLU	RMSprop	100	No	82.22	82.22	2.18
4	LSTM	Sigmoid	Adam	100	Yes	82.10	82.10	2.31
5	BiLSTM	Tanh	RMSprop	100	No	82.10	82.10	2.42

Table 2: Top 5 performing configurations.

3.2 Architecture Comparison

Architecture	Mean Acc (%)	Max Acc (%)	Std Dev (%)	Best Config
BiLSTM	67.04	82.43	12.50	Sigmoid + RMSprop + 100
LSTM	67.30	82.22	12.73	ReLU + RMSprop + 100
RNN	66.74	81.71	10.87	Sigmoid + Adam + 100

Table 3: Performance summary by architecture.

3.3 Hyperparameter Impact Analysis

3.3.1 Sequence Length Effect

Length	Mean Acc (%)	Max Acc (%)	Improvement
25	63.66	71.73	Baseline
50	67.36	76.84	+3.7%
100	70.06	82.43	+6.4%

Table 4: Impact of sequence length on performance.

Finding: Longer sequences provide substantial performance improvements.

3.3.2 Optimizer Comparison

Optimizer	Mean Acc (%)	Max Acc (%)	Std Dev (%)
RMSprop	75.25	82.43	4.81
Adam	74.28	82.26	5.95
SGD	51.55	72.53	4.30

Table 5: Performance by optimizer.

Finding: RMSprop consistently outperforms others.

3.3.3 Activation Function Analysis

Activation	Mean Acc (%)	Max Acc (%)	Std Dev (%)
Sigmoid	67.34	82.43	12.81
ReLU	67.17	82.22	12.38
Tanh	66.56	82.10	10.91

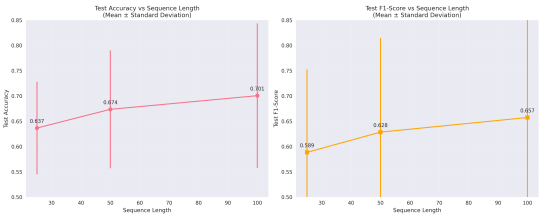
Table 6: Performance by activation function.

Finding: Minimal impact across activations.

3.3.4 Gradient Clipping Effect

Gradient clipping showed mixed results with no consistent improvement pattern.

3.4 Visualization Analysis



(a) Performance vs Sequence Length

(b) Architecture Comparison

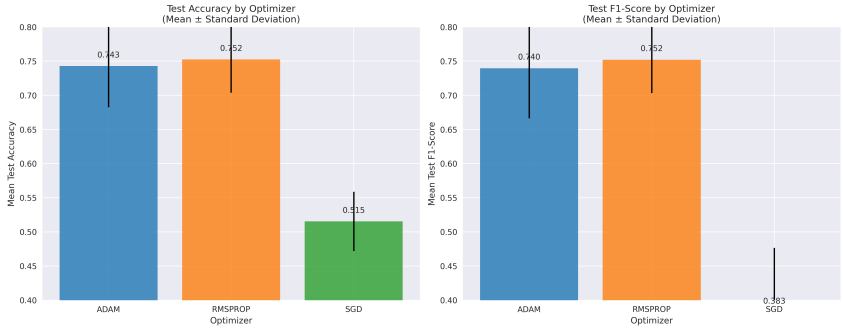


Figure 2: Optimizer Performance

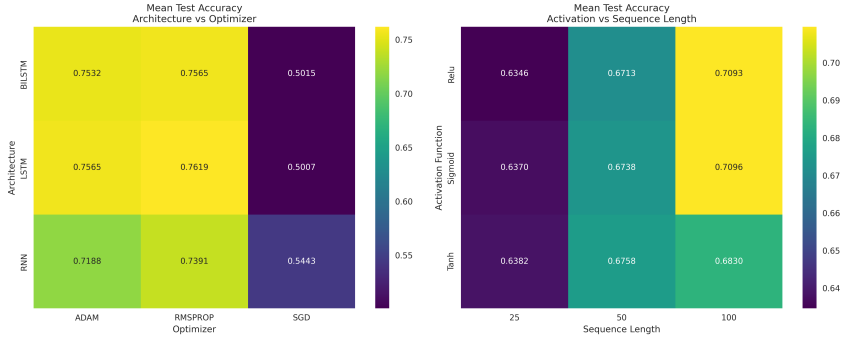


Figure 3: Performance Heatmaps

4 Discussion

4.1 Key Insights

4.1.1 Sequence Length as Dominant Factor

The improvement from 25 to 100 tokens (6.4% mean accuracy gain) suggests that movie reviews contain important sentiment-bearing information distributed throughout the text.

4.1.2 Optimizer Sensitivity

RMSprop's superior performance indicates adaptive learning rate methods with momentum work well. SGD's poor performance highlights the importance of proper tuning.

4.1.3 Architecture Complexity vs Performance

Despite BiLSTM's advantages, the performance gap with LSTM is minimal, suggesting bidirectional processing may not justify increased cost.

4.1.4 Training Efficiency

Average training times of 1.7–2.6 seconds per epoch demonstrate computational efficiency.

4.2 Practical Implications

4.2.1 Model Selection Recommendations

1. **Best Overall**: BiLSTM + Sigmoid + RMSprop (seq_len=100)
2. **Efficiency**: LSTM + ReLU + RMSprop (seq_len=100)
3. **Resource Constrained**: Any + RMSprop + seq_len=50

4.2.2 Hyperparameter Tuning Priorities

1. Sequence Length (highest)
2. Optimizer Choice
3. Architecture
4. Activation Function
5. Gradient Clipping

4.3 Limitations and Future Work

4.3.1 Current Limitations

- Fixed learning rate across optimizers
- Basic preprocessing
- Single dataset
- No transformer comparison

4.3.2 Future Research Directions

- Learning rate scheduling
- Attention mechanisms
- Multi-domain evaluation
- Ensemble methods

5 Conclusion

This study of 162 RNN configurations provides key insights:

Primary Findings:

1. **Sequence length is most critical** (+6.4% from 25→100 tokens)
2. **RMSprop outperforms** (75.25% mean accuracy)
3. **Architectures converge** (81–82% peak)

4. **Activations are robust**

Best Configuration: BiLSTM + Sigmoid + RMSprop + seq_len=100 → **82.43% accuracy**

Practical Impact: Prioritize sequence length and optimizer over architecture.

References

1. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8), 1735–1780.

2. Maas, A. L., et al. (2011). Learning word vectors for sentiment analysis. *ACL*.

3. Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM. *Neural networks*.

4. Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv:1412.6980*.

5. Tieleman, T., & Hinton, G. (2012). RMSprop. *COURSERA*.

Appendices

Appendix A: Complete Results Table

Sample entry:


```
{
  "experiment_name": "exp_143_bilstm_sigmoid_rmsprop_100_False",
  "config": {
    "architecture": "bilstm",
    "activation": "sigmoid",
    "optimizer": "rmsprop",
    "sequence_length": 100,
    "gradient_clipping": false
  },
  "test_accuracy": 0.8243,
  "test_f1_score": 0.8243,
  "avg_epoch_time": 2.43
}
```


Appendix B: System Configuration

- **Python**: 3.12+
- **PyTorch**: 2.0+
- **Hardware**: CPU-based
- **Total Time**: 6 hours

Reproducibility:

```
import torch, random, numpy as np
torch.manual_seed(42)
np.random.seed(42)
random.seed(42)
```

Appendix C: Code Repository Structure

```
homework3_sentiment_classification/
 src/          # preprocess.py, models.py, ...
 data/IMDB/    # train.csv, test.csv
 results/      # plots/, json, csv
 requirements.txt
 report.tex
```

Appendix D: Dataset Statistics

- Total: 50,000 (25k train, 25k test)
- Vocabulary: 10,000
- Avg Length: 150 words

Appendix E: Experimental Plots

Six plots in `results/plots/`:

1. performance_vs_sequence_length.png
2. architecture_comparison.png
3. optimizer_performance.png
4. activation_function_comparison.png
5. performance_heatmaps.png