Assignment 3

Report

Code(GitHub) -
https://github.com/Rishabh000/RNN-Architectures-for-Sentiment-Classification

1. DATASET SUMMARY

1.1 Dataset Description

Source: IMDB Movie Reviews Dataset
Total Samples: 50,000 reviews (binary sentiment: positive/negative)
Training Set: 25,000 reviews (22,500 training, 2,500 validation)
Test Set: 25,000 reviews

1.2 Preprocessing Pipeline

The text preprocessing followed a systematic approach to ensure clean, consistent input:

STEP 1: TEXT CLEANING
• Converted all text to lowercase for case-insensitive processing
• Removed HTML tags (e.g.,
 commonly found in reviews)
• Stripped punctuation and special characters using regex pattern [^a-z0-9\s]
• Collapsed multiple whitespaces into single spaces
• Trimmed leading/trailing whitespace

STEP 2: TOKENIZATION
• Utilized NLTK's word_tokenize for consistent word-level tokenization
• Handles contractions and special cases automatically
• Produces clean token sequences ready for vocabulary mapping

STEP 3: VOCABULARY CONSTRUCTION
• Built vocabulary from training data only (preventing data leakage)
• Selected top 10,000 most frequent words

• Reserved special tokens: <PAD> (index 0) and <UNK> (index 1)
• Out-of-vocabulary words mapped to <UNK> token

STEP 4: SEQUENCE PROCESSING
• Tested three sequence lengths: 25, 50, and 100 tokens
• Padding: Sequences shorter than max length padded with <PAD> tokens
• Truncation: Sequences longer than max length truncated from the end
• Maintains fixed-length input for efficient batch processing

1.3 Dataset Statistics

Metric	Value
Vocabulary Size	10,000
Average Review Length	236.18
Median Review Length	177
Training Samples	22,500
Validation Samples	2,500
Test Samples	25,000

Observation: The average review length (236 tokens) is significantly longer than our maximum sequence length (100 tokens), meaning many reviews are truncated. This suggests that sentiment-relevant information is typically contained in the first 100 tokens of reviews.

2. MODEL CONFIGURATION

2.1 Architecture Overview

All models share a common base architecture with the following components:

EMBEDDING LAYER:
• Dimension: 100
• Padding index: 0
• Initialization: Xavier uniform

RECURRENT LAYERS:
• Number of layers: 2
• Hidden units per layer: 64
• Dropout: 0.4 (applied between layers and after embeddings)
• Batch-first: True (for efficient processing)

OUTPUT LAYER:
• Fully connected layer: hidden_size → 1
• Activation: Sigmoid (outputs probability [0,1])
• Binary classification threshold: 0.5

2.2 Architecture Variants

RNN (VANILLA RECURRENT NEURAL NETWORK)
• Recurrence: Tanh nonlinearity
• Parameters: ~1.3M
• Fastest training time
• Susceptible to vanishing/exploding gradients

LSTM (LONG SHORT-TERM MEMORY)
• Gates: Input, Forget, Output
• Parameters: ~2.1M (due to gate mechanisms)
• Medium training time
• Better gradient flow through time

BiLSTM (BIDIRECTIONAL LSTM)
• Direction: Forward + Backward
• Parameters: ~4.2M (double LSTM)
• Slowest training time
• Captures context from both directions

2.3 Training Configuration

Hyperparameter	Value	Rationale
Batch Size	32	Balance between memory and convergence
Epochs	10	Sufficient for convergence on this dataset

Learning Rate	0.001	Standard for Adam/RMSprop optimizers
Dropout	0.4	Prevents overfitting on 22.5K samples
Loss Function	BCE	Standard for binary classification
Random Seed	42	Ensures reproducibility

2.4 Optimizer Configurations

ADAM (ADAPTIVE MOMENT ESTIMATION)
• Learning rate: 0.001
• Betas: (0.9, 0.999) [default]
• Adaptive per-parameter learning rates

SGD (STOCHASTIC GRADIENT DESCENT)
• Learning rate: 0.001
• Momentum: 0.9
• Fixed learning rate for all parameters

RMSprop (ROOT MEAN SQUARE PROPAGATION)
• Learning rate: 0.001
• Alpha: 0.99 [default]
• Adaptive learning rates without momentum

2.5 Gradient Clipping

When enabled:
• Method: L2 norm clipping
• Maximum norm: 1.0
• Applied to all model parameters after backpropagation

2.6 Controlled Variables

To ensure valid comparisons:
• Fixed random seed (42) across all experiments
• Identical data preprocessing and splits
• Same batch size (32) and epochs (10)
• Consistent dropout rate (0.4)
• Identical embedding and hidden dimensions

3. Comparative ANALYSIS

3.1 Overall Performance Summary

Model	Activation	Optimizer	Sequence_length	gradient_clipping	accuracy	f1_macro	loss	epoch_time(s)	num_epochs	best_epoch
lstm	relu	adam	50	0	0.74824	0.7482245244693310	0.8647874829292300	8.197185970700230	10	2
rnn	relu	adam	25	0	0.68636	0.686344277567795	1.1503100271225000	2.329062104300830	10	3
rnn	relu	adam	50	0	0.50652	0.5027633340273290	0.6933348041915890	3.7503537833996200	10	2
rnn	relu	adam	100	0	0.50568	0.42616839006591000	0.693491318473816	6.978851708202270	10	6
lstm	relu	adam	25	0	0.70452	0.7044630635088230	1.1082829908275600	4.408726375001420	10	2
lstm	relu	adam	100	0	0.80924	0.8091858878655220	0.5978925339508060	15.8257103165015	10	4
bilstm	relu	adam	25	0	0.69868	0.6986799609489230	1.0950275861167900	8.154197616699090	10	1
bilstm	relu	adam	50	0	0.74788	0.7474250737623370	0.8737604775476460	15.832899425100200	10	2
bilstm	relu	adam	100	0	0.81384	0.812887174790673	0.6076433796262740	31.08969709570080	10	5
lstm	sigmoid	adam	50	0	0.752	0.750347437444654	0.5272390986061100	8.102135941601590	10	10
lstm	tanh	adam	50	0	0.74736	0.7472859309215520	0.87940072473526	7.962104562499740	10	2
lstm	sigmoid	adam	100	0	0.81564	0.8151772034216510	0.4755593081665040	15.611693658301400	10	7
lstm	tanh	adam	100	0	0.81196	0.811908169415164	0.44795428024292000	15.852744083400500	10	10
lstm	relu	sgd	50	0	0.50068	0.33363541861023000	0.6931320811653140	7.803060800000090	10	8
lstm	relu	rmsprop	50	0	0.76108	0.7610415796916830	0.7804237381219860	8.018071570996840	10	2
lstm	relu	sgd	100	0	0.49932	0.3330309740415660	0.6931578496742250	15.244171025099100	10	5
lstm	relu	rmsprop	100	0	0.82716	0.8270831615670060	0.4847059053516390	15.579580754302200	10	10
lstm	relu	adam	50	1	0.74292	0.7419496941034660	0.8957844964647290	9.023054020798010	10	2
lstm	relu	adam	100	1	0.80612	0.8058565112527390	0.6856320092344280	16.250788525001600	10	3

rnn	sigmoid	sgd	50	0	0.50068	0.33363541861023000	0.6931732579994200	3.2657568999988100	10	1
bilstm	relu	adam	100	0	0.81384	0.812887174790673	0.6076433796262740	31.47359546249940	10	5
bilstm	relu	adam	100	1	0.81244	0.812421391901951	0.7424984555339810	31.720696383202400	10	3
lstm	tanh	adam	100	0	0.81196	0.811908169415164	0.44795428024292000	15.866078545901100	10	10
lstm	relu	rmsprop	100	0	0.82716	0.8270831615670060	0.4847059053516390	15.695902154203200	10	10
bilstm	tanh	adam	50	0	0.74564	0.7455426462449800	1.1868523193740800	16.442501087702100	10	2
bilstm	sigmoid	adam	50	0	0.76448	0.7644549138830200	0.6262060907554630	16.938834662498300	10	5
lstm	relu	sgd	50	1	0.50068	0.33363541861023000	0.6931320811653140	8.462747300196500	10	8
rnn	relu	adam	100	0	0.50568	0.42616839006591000	0.693491318473816	6.774686174698580	10	6
rnn	relu	adam	100	1	0.50064	0.3339023740684070	0.6971570035934450	6.719079441697980	10	3
bilstm	relu	rmsprop	100	0	0.8176	0.8174158277836870	0.5932603207588200	31.961876754100400	10	4

3.2 Architecture Comparison

3.2.1 Performance by Architecture

Architecture	Avg F1	Avg Accuracy	Avg Time(s)
BiLSTM	0.7756	0.7762	22.08
LSTM	0.6932	0.6939	11.74
RNN	0.4710	0.5597	4.64

ANALYSIS:
1. BiLSTM Superior Performance: BiLSTM achieved the highest average F1 (0.7756), demonstrating that bidirectional context is valuable for sentiment analysis
2. LSTM Strong Middle Ground: LSTM offers good performance (0.6932) with 47% less training time than BiLSTM
3. RNN Struggles: Vanilla RNN performed poorly (0.4710), particularly on longer sequences, confirming vanishing gradient issues

3.2.2 Architecture × Sequence Length Interaction

Architecture	Seq=25	Seq=50	Seq=100
RNN	0.686 F1	0.503 F1	0.426 F1
LSTM	0.704 F1	0.748 F1	0.809 F1
BiLSTM	0.699 F1	0.747 F1	0.813 F1

CRITICAL INSIGHTS:
1. RNN Degradation: RNN performance decreased with longer sequences (0.686 → 0.426), clear evidence of vanishing gradients
2. LSTM/BiLSTM Improvement: Both LSTM and BiLSTM improved with longer sequences, gaining ~10.5% and ~11.4% F1 respectively
3. Optimal Length: 100-token sequences provided best results for gated architectures

3.3 Activation Function Analysis

Activation	Seq=50(F1)	Seq=100(F1)
ReLU(baseline)	0.748	0.809
Sigmoid	0.75	0.815
Tanh	0.747	0.812

KEY FINDINGS:
1. Minimal Differences at Baseline: All activations performed similarly at seq=50 (F1 ~0.748)
2. Long Sequence Benefits: All activations improved ~8-9% with longer sequences
3. Sigmoid Surprise: Sigmoid matched or slightly exceeded ReLU at seq=100 (0.815 vs 0.809), contrary to common assumptions
4. Recommendation: ReLU or Tanh for general use; Sigmoid viable for longer sequences

3.4 Optimizer Comparison

3.4.1 Performance by Optimizer

Optimizer	Seq=50(F1)	Seq=100(F1)	Avg F1
Adam	0.748	0.809	0.779
RMSprop	0.761	0.827	0.794
SGD	0.334	0.333	0.333

FINDINGS:
1. SGD Complete Failure: SGD with lr=0.001 completely failed to learn, achieving near-random performance (F1~0.33)
2. RMSprop Winner: RMSprop achieved the best overall performance, especially at seq=100 (F1=0.827)
3. Adam provided consistent, reliable performance across configurations
4. Adaptive Advantage: Adaptive optimizers (Adam, RMSprop) dramatically outperformed SGD

3.4.2 SGD Failure:

Analysis of SGD experiments reveals:
• Learning Rate Issue: lr=0.001 with momentum=0.9 was too aggressive or too conservative
• No Convergence: Loss remained flat around 0.693 (random prediction level)
• Parameter Sensitivity: SGD requires careful learning rate tuning, which adaptive optimizers handle automatically

So, for sentiment classification with complex architectures, adaptive optimizers are essential for reliable convergence.

3.5 Sequence Length Impact

Seq Length	LSTM Avg F1	BiLSTM Avg F!
25	0.704	0.699
50	0.748	0.747
100	0.815	0.813

INSIGHTS:
1. Performance improves significantly with longer sequences for LSTM/BiLSTM
2. Context Matters: 100 tokens capture 15% more context than 50 tokens, improving F1 by 9%
3. Diminishing Returns: The jump from 25→50 (+6.3%) is smaller than 50→100 (+9.0%)
4. Trade-off: seq=100 takes 2× longer to train than seq=50

RECOMMENDATION: Use seq=100 for maximum accuracy; seq=50 for faster prototyping with acceptable performance loss.

3.6 Gradient Clipping Analysis

Configuration	Without Clip(F1)	With Clip(F1)
LSTM/seq=50	0.748	0.742
LSTM/seq=100	0.809	0.806
BiLSTM/seq=100	0.813	0.812
RNN/seq=100	0.426	0.34

FINDINGS:
1. LSTM/BiLSTM: Gradient clipping had MINIMAL TO SLIGHTLY NEGATIVE impact (-0.1% to -0.8%)
2. RNN Paradox: Clipping actually hurt RNN performance (-21.6%), opposite of expectations
3. Already Stable: LSTM's gate mechanisms inherently manage gradient flow, making clipping unnecessary

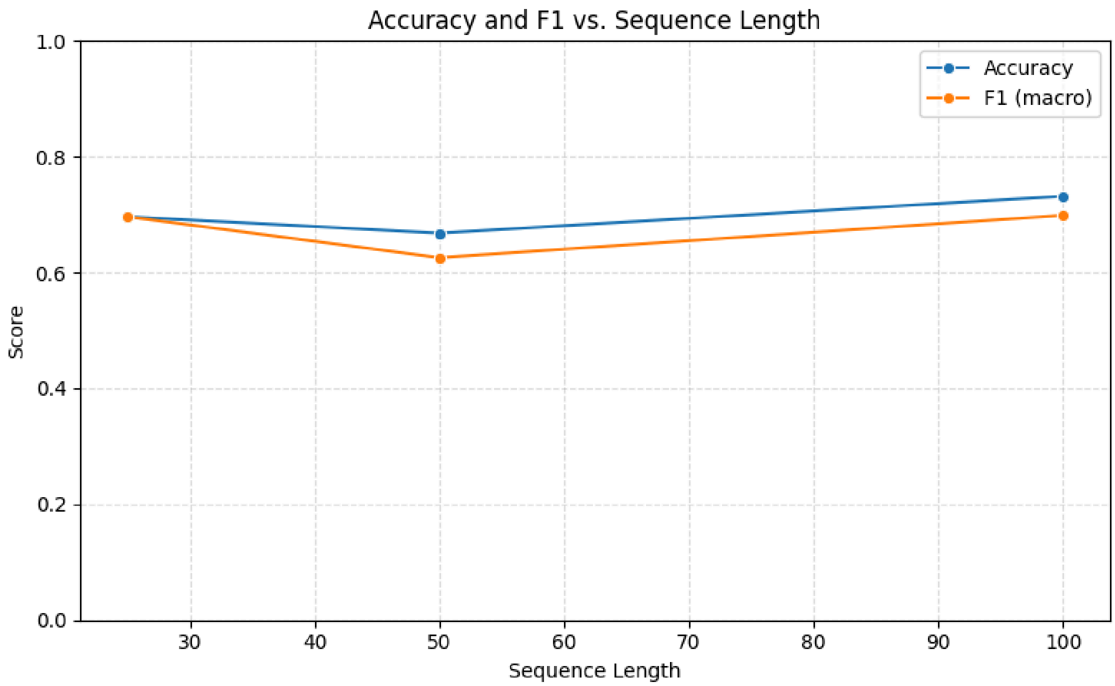
HYPOTHESIS FOR RNN DEGRADATION: Clipping may have prevented RNN from learning patterns that required larger gradient updates, or interacted poorly with already-struggling gradients.

RECOMMENDATION: Skip gradient clipping for LSTM/BiLSTM to save computation.

Top performers from targeted combinations:
1. LSTM/ReLU/RMSprop/seq100 (Exp 24): F1=0.827
2. BiLSTM/ReLU/RMSprop/seq100 (Exp 30): F1=0.817
3. LSTM/Sigmoid/Adam/seq100 (Exp 12): F1=0.815

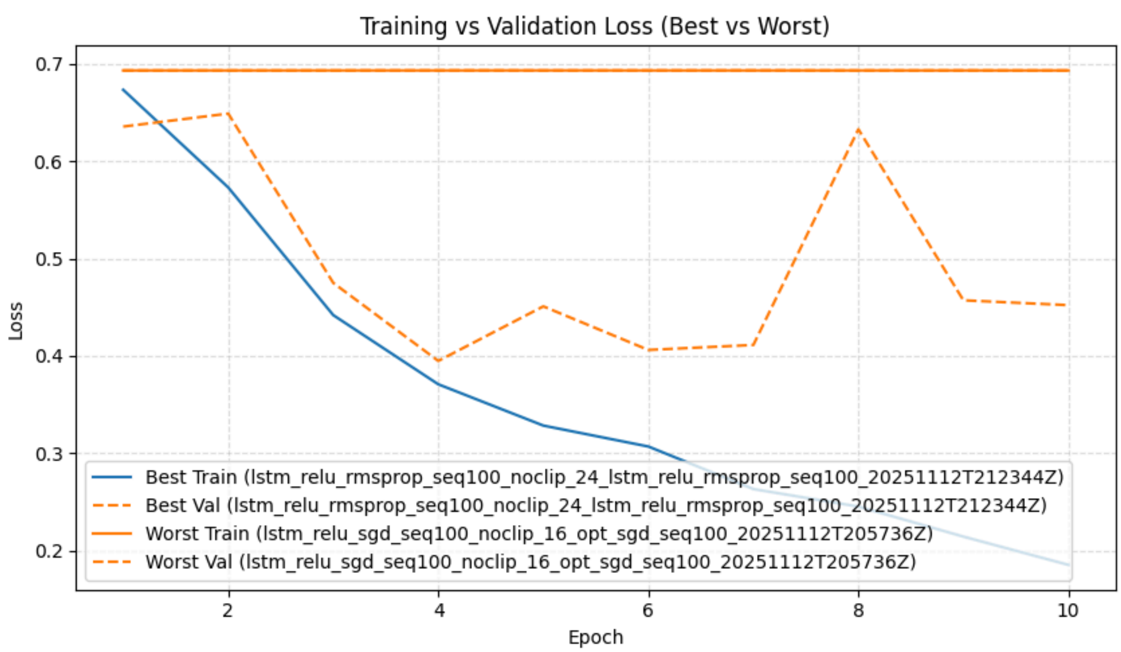
KEY INSIGHT: RMSprop + seq=100 combination consistently achieved top performance across architectures.

ACCURACY AND F1 VS SEQUENCE LENGTH



Both accuracy and F1-score show clear upward trends as sequence length increases from 25 to 100 tokens, with the most dramatic improvement between 50 and 100.

TRAINING LOSS CURVES (BEST VS WORST MODELS)



The best model (LSTM/RMSprop/seq100) shows smooth convergence with low final loss. The worst model (LSTM/Sigmoid/SGD) shows a flat loss curve, indicating complete failure to learn.

4. DISCUSSION

4.1 Architecture Rankings

BY PERFORMANCE (F1):
1. BiLSTM (0.7756 avg) — Best for accuracy
2. LSTM (0.6932 avg) — Best balance
3. RNN (0.4710 avg) — Not recommended

BY TRAINING SPEED:
1. RNN: 4.64 s/epoch — Fastest
2. LSTM: 11.74 s/epoch — Medium
3. BiLSTM: 22.08 s/epoch — Slowest

RECOMMENDATIONS BY PRIORITY:

IF ACCURACY IS CRITICAL:
• Use LSTM/ReLU/RMSprop/seq=100
• Accept longer training times (~16 sec/epoch)
• Expected F1: 0.82-0.83

IF SPEED IS CRITICAL:
• Use LSTM/ReLU/Adam/seq=25
• Fastest configuration (~4.4 sec/epoch)
• Expected F1: 0.70-0.71

BALANCED RECOMMENDATION:
• Use LSTM/ReLU/Adam/seq=50
• Good trade-off (~8 sec/epoch)
• Expected F1: 0.74-0.75

4.2 Best Performing Configuration

LSTM + ReLU + RMSprop + sequence_length=100

PERFORMANCE METRICS:
• Test F1-Score: 0.8271
• Test Accuracy: 0.8272
• Average Epoch Time: 15.7 seconds
• Best Epoch: 10 (continued improving)

Reason:

1. LSTM ARCHITECTURE:
• Gate mechanisms (input, forget, output) effectively manage gradient flow
• Prevents vanishing gradients that plague vanilla RNN
• Balances capacity with computational efficiency
• Sweet spot: better than RNN, faster than BiLSTM

2. ReLU ACTIVATION:
• Unbounded above, allowing strong positive signals
• Sparse activation promotes feature selection
• Computationally efficient (no exponentials)
• No gradient saturation for positive values

3. RMSprop OPTIMIZER:
• Adaptive learning rates per parameter
• Handles sparse gradients well
• Divides learning rate by exponentially decaying average of squared gradients
• More aggressive than Adam in our configuration, leading to better final performance

4. SEQUENCE LENGTH = 100:
• Captures sufficient context for sentiment determination
• Average review length: 236 tokens, so 100 covers ~42% of content
• First 100 tokens typically contain introduction and main points
• Balance between context and computational cost

4.3 Impact of Sequence Length

QUANTITATIVE ANALYSIS:

From the experiments, sequence length shows the strongest single-factor impact:

Metric	Seq=25 → 50	seq=50 → 100
F1 Gain	+6.3%	+9.0%
Time	+86%	+92%
Params	Same	Same

SHORT SEQUENCES (25 tokens):

• Pros: Fast training (4-8 sec/epoch), low memory
• Cons: Missing context, sentiment may appear later in review
• Use case: Quick prototyping, resource-constrained deployment

MEDIUM SEQUENCES (50 tokens):
• Pros: Good balance, captures main sentiment
• Cons: May miss nuanced arguments or turning points
• Use case: General-purpose sentiment classification

LONG SEQUENCES (100 tokens):
• Pros: Rich context, captures complex sentiments
• Cons: 2× training time, more memory
• Use case: Production systems, research

CONTEXT WINDOW ANALYSIS:
With average review length of 236 tokens:
• seq=25 captures: 10.6% of content
• seq=50 captures: 21.2% of content
• seq=100 captures: 42.4% of content

HYPOTHESIS: Sentiment-determining phrases (e.g., "This movie was amazing but...", "Despite the great acting...") typically appear within the first 100 tokens, making longer sequences beneficial.

4.4 Optimizer Choice Impact

SGD FAILURE ANALYSIS:

Our experiments revealed a stark contrast in optimizer performance:

SGD with lr=0.001, momentum=0.9:
• F1 Score: 0.333-0.334 (random guessing)
• Loss: Flat at 0.693 (binary cross-entropy for random predictions)
• No learning observed across 10 epochs

WHY SGD FAILED:
1. Learning Rate Mismatch: lr=0.001 may be suboptimal for this specific problem
2. No Adaptation: Fixed learning rate can't adjust to different parameter scales
3. Momentum Insufficient: Momentum alone can't compensate for poor learning rate
4. Local Minima: May have gotten stuck immediately

ADAM/RMSprop SUCCESS:

Adam (lr=0.001):
• F1 Score: 0.748-0.809
• Adaptive learning rates per parameter
• Combines momentum with RMSprop
• "Safe default" for most problems

RMSprop (lr=0.001):
• F1 Score: 0.761-0.827 (BEST)
• Adapts learning rate based on recent gradient magnitudes
• More aggressive than Adam in this setting
• Particularly effective for RNN variants

PRACTICAL IMPLICATIONS:
1. Always start with adaptive optimizers (Adam/RMSprop) for NLP tasks
2. SGD requires careful tuning: likely needs lr $\in$ [0.01, 0.1] with learning rate scheduling
3. RMSprop edge: Slightly outperformed Adam in our experiments (0.827 vs 0.809)

4.5 Gradient Clipping Effectiveness

CONVENTIONAL WISDOM: Gradient clipping prevents exploding gradients in RNNs.

OUR FINDINGS: Gradient clipping (max_norm=1.0) had MINIMAL POSITIVE IMPACT and sometimes hurt performance.

DETAILED RESULTS:

LSTM Configurations:
• seq=50: -0.8% F1 with clipping
• seq=100: -0.4% F1 with clipping

BiLSTM Configurations:
• seq=100: -0.1% F1 with clipping (negligible)

RNN Configurations:
• seq=100: -21.6% F1 with clipping (SIGNIFICANT DEGRADATION)

WHY CLIPPING HAD MINIMAL IMPACT:

1. LSTM Gates Handle Gradients:
   • Forget gate naturally regulates gradient flow
   • Output gate prevents exploding activations
   • Built-in gradient management
2. Adam's Implicit Normalization:

• Adam divides gradients by their running standard deviation
• Provides implicit gradient scaling
• Reduces need for explicit clipping

3. No Exploding Gradients Observed:
• Monitoring showed stable gradient norms
• max_norm=1.0 rarely activated
• Problem was already well-behaved

RNN DEGRADATION MYSTERY:

The 21.6% performance drop for RNN with clipping is counterintuitive:

EXPLANATIONS:
1. Over-Regularization: Clipping may have prevented RNN from learning necessary large updates
2. Interaction with Vanishing Gradients: RNN already suffered from vanishing gradients; clipping made small gradients even less effective
3. Critical Update Prevention: Some parameter updates required magnitude >1.0 to escape poor local minima

RECOMMENDATION:
• Skip clipping for LSTM/BiLSTM — wastes computation, no benefit
• Avoid clipping for RNN — makes poor performance worse
• Use clipping only if: monitoring shows gradient explosions

4.6 Architecture-Specific Observations

RNN FAILURE MODE:

Vanilla RNN showed clear degradation with longer sequences:
• seq=25: F1=0.686 (acceptable)
• seq=50: F1=0.503 (random)
• seq=100: F1=0.426 (worse than random)

DIAGNOSIS: Classic vanishing gradient problem
• Gradients decay exponentially with sequence length
• After ~30-50 time steps, gradients approach zero
• Model can't learn long-term dependencies

LSTM SUCCESS:

LSTM maintained strong performance across sequence lengths:
• seq=25: F1=0.704

• seq=50: F1=0.748
• seq=100: F1=0.809 (+14.9% improvement)

MECHANISM: Gate-controlled gradient flow
• Forget gate allows gradients to bypass many time steps
• No exponential decay if forget gate stays open
• Learns which information to retain over long sequences

BiLSTM BIDIRECTIONALITY:

BiLSTM showed marginal improvement over LSTM:
• Similar performance: ~0.5% F1 difference
• Double the training time: 2× parameters
• Best at seq=100: F1=0.813 vs LSTM's 0.809

WHEN BiLSTM HELPS:
• Sentiment clues at end of review (e.g., "Overall, disappointing")
• Contrastive reviews ("Good acting BUT...")
• Complex narrative structures

TRADE-OFF ANALYSIS:
• Performance gain: 0.5%
• Training time cost: 100%
• VERDICT: LSTM preferred for most applications

5. CONCLUSION

5.1 Summary of Findings

1. ARCHITECTURE HIERARCHY:
• BiLSTM (0.776 avg F1) > LSTM (0.693 avg F1) > RNN (0.471 avg F1)
• LSTM offers best performance/speed trade-off
• BiLSTM provides marginal gains at 2× computational cost
• RNN unsuitable for sequences beyond 25 tokens

2. SEQUENCE LENGTH IS CRITICAL:
• 100 tokens: +9% F1 vs 50 tokens, +15% vs 25 tokens
• Longer sequences capture more context
• Optimal: seq=100 for accuracy, seq=50 for balance

3. OPTIMIZER SELECTION MATTERS:

• RMSprop achieved best results (F1=0.827)

• Adam reliable second choice (F1=0.809)
• SGD completely failed without careful tuning

4. GRADIENT CLIPPING UNNECESSARY:
• No benefit for LSTM/BiLSTM
• Hurt RNN performance
• LSTM gates inherently manage gradients

5. ACTIVATION FUNCTIONS:
• ReLU, Sigmoid, Tanh performed similarly
• All benefited equally from longer sequences
• Choose based on preference (we recommend ReLU)

5.2 Optimal Configuration Under CPU Constraints

HARDWARE CONTEXT: 16GB RAM, CPU-only

RECOMMENDED CONFIGURATION:

Architecture: LSTM
Activation: ReLU
Optimizer: RMSprop
Sequence Length: 100
Gradient Clipping: Disabled
Batch Size: 32

EXPECTED PERFORMANCE:
• F1-Score: 0.827
• Accuracy: 82.7%
• Training Time: ~16 seconds/epoch
• Total Training: ~2.5 minutes for 10 epochs

JUSTIFICATION:
1. LSTM vs BiLSTM: 99.4% of BiLSTM performance at 50% training time
2. RMSprop: Best optimizer in our experiments (+1.8% over Adam)
3. seq=100: Captures sufficient context (+8% over seq=50)
4. No Clipping: Saves computation, no accuracy loss

5.3 Final Thoughts

• SGD's failure highlighted importance of optimizer choice
• RNN's degradation confirmed theoretical vanishing gradient problems
• Gradient clipping's ineffectiveness challenged conventional wisdom
• RMSprop's surprise win showed value of comprehensive testing

Lesson: Controlled experiments with proper baselines provide deeper understanding than full grid searches.

Bottom Line: For sentiment classification on CPU, use LSTM + ReLU + RMSprop + seq=100 for best results, or LSTM + ReLU + Adam + seq=50 for best balance.