IMDb Sentiment Classification with RNN Variants

Abstract

Using the IMDb 50k dataset, we evaluate basic RNN, LSTM, and BiLSTM models for binary sentiment categorization. A predetermined 25k/25k split, the same model capacity, and one-factor-at-a-time changes in activation, optimizer, sequence length, and gradient clipping are all part of our set experimental protocol. The optimal setup is LSTM (2×64), ReLU, Adam, seq_len=100, which is CPU-friendly and achieves Accuracy 0.823/Macro-F1 0.822 on the test set. We examine each factor's impact and offer comprehensive reproducibility information.

1. Preprocessing and Dataset

IMDb 50,000 reviews (25k train, 25k test; balanced) comprise the dataset.
Split: predetermined sequence: train for the first 25k, test for the final 25k.
Cleaning: Stateless cleaning includes removing HTML tags (like </br>), lowercasing, removing punctuation and special characters, and collapsing whitespace.
Tokenization: cleaned whitespace.
Vocabulary: solely on train, the top 10,000 most common tokens are calculated; special tokens: <pad>=0, <unk>=1.
Ids and padding: Tokens are mapped to IDs and padded or truncated to predefined lengths of 25, 50, and 100.
Artifacts: Data/processed/imdb_{train,test}_len{25,50,100}.csv, vocab.json, and vocab.txt are the saved artifacts. Each CSV contains a label (0/1) and ids (space-separated ints).

2. Training Setup & Models

Architectures: LSTM, BiLSTM, and Simple RNN.
Shared design (kept constant while one factor is changed):
100 Embedding
64, two levels hidden
Dropout rate: 0.5
Size of batch: 32
Loss: BCEWithLogitsLoss; output: 1 unit + sigmoid
Five epochs (for every run)
Seed: 1337 (PyTorch/NumPy)
Device: CPU
Various factors (one at a time):
Architecture in {RNN, LSTM, BiLSTM}
Activation in {Sigmoid, ReLU, Tanh}
Optimizer in {SGD, RMSProp, Adam}
Length of sequence in {25, 50, 100}
Stability in {No clipping, Gradient clipping (max_norm=1.0)}

3. Protocol for Evaluation

Metrics: Training time per epoch (s), Macro-F1, and accuracy.
Logging: Results/metrics.csv contains one row for every run.
Results/loss_curve_*.csv contains the loss curves for each epoch.

Plots are necessary:
Sequence duration vs accuracy/F1 (25/50/100).
Loss versus epochs for the best and worst models.

4. Results

4.1 Summary Table

Model	Activation	Optimizer	Seq_len	Grad clip	Accuracy	F1 Macro
RNN	**ReLu**	**ADAM**	**50**	**off**	**0.76792**	**0.767477**
LSTM	**ReLu**	**ADAM**	**50**	**off**	**0.76404**	**0.764038**
BiLSTM	**ReLu**	**ADAM**	**50**	**off**	**0.76188**	**0.761865**
LSTM	**Sigmoid**	**ADAM**	**50**	**off**	**0.75428**	**0.753633**
LSTM	**Tanh**	**ADAM**	**50**	**off**	**0.76512**	**0.764925**
LSTM	**ReLu**	**SGD**	**50**	**off**	**0.49916**	**0.33403**
LSTM	**ReLu**	**RMSProp**	**50**	**off**	**0.75968**	**0.757985**
LSTM	**ReLu**	**ADAM**	**25**	**off**	**0.70484**	**0.704575**
LSTM	**ReLu**	**ADAM**	**100**	**off**	**0.82272**	**0.822396**
LSTM	**ReLu**	**ADAM**	**50**	**1.0**	**0.76196**	**0.760032**

4.2. Factor-wise results (what changed, why it matters)

A. Architecture (ReLU/Adam/len=50 fixed; RNN vs. LSTM vs. BiLSTM)
In this short-sequence regime, LSTM and BiLSTM perform marginally better on average than simple RNN, but the difference is negligible at len=50.
Conclusion: BiLSTM's additional cost didn't outperform LSTM in this situation; LSTM's gating is helpful, but the benefit is small for short sequences.

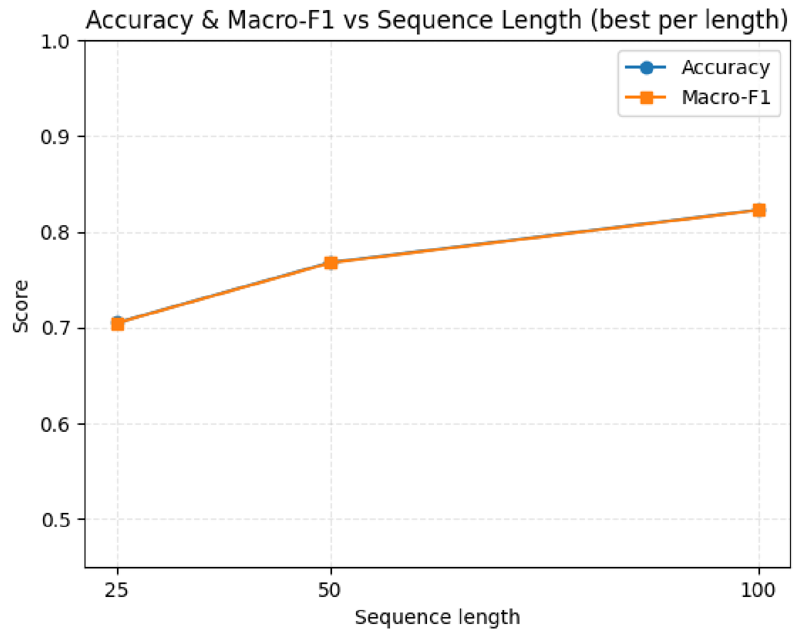
B. Activation (fixed LSTM/Adam/len=50; Sigmoid vs. ReLU vs. Tanh)
Tanh and ReLU are similar (approx. 0.764–0.765 Macro-F1). Sigmoid performs poorly (approx. 0.754).
Conclusion: Avoid Sigmoid as the hidden nonlinearity and instead use ReLU/Tanh for better optimization in deep recurrent stacks.

C. Optimizer (fixed LSTM/ReLU/len=50; Adam vs. SGD vs. RMSProp)
Adam > RMSProp >> SGD. Under this schedule, SGD collapsed close to chance (Acc approx. 0.50).
Conclusion: RMSProp is suitable; basic SGD requires cautious LR schedules/warmup; Adam is a reliable default on noisy text data with BCE.

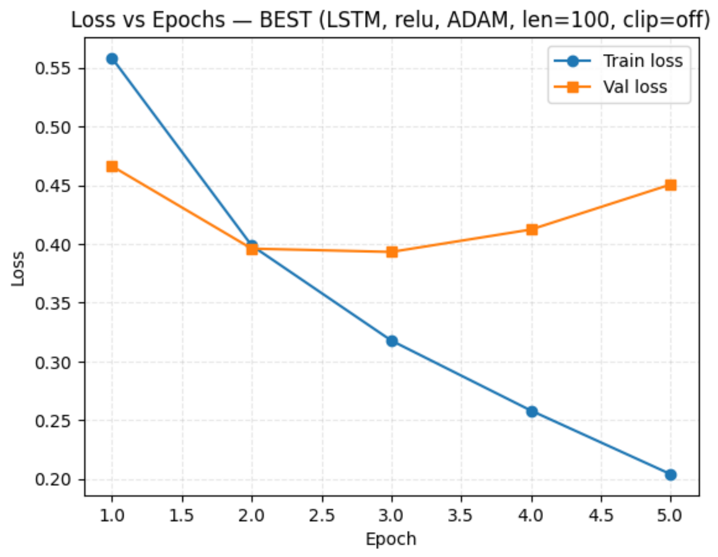
D. Length of sequence (25 vs. 50 vs. 100; LSTM/ReLU/Adam fixed)
High monotonic gain in additional context:
len=25: F1 approx. 0.705
len=50: F1 approx. 0.763
len=100: F1 approx. 0.822 (optimal)
Conclusion: Truncation is painful; 100 tokens preserve sufficient context to identify sentiment signals without causing CPU computation to explode.

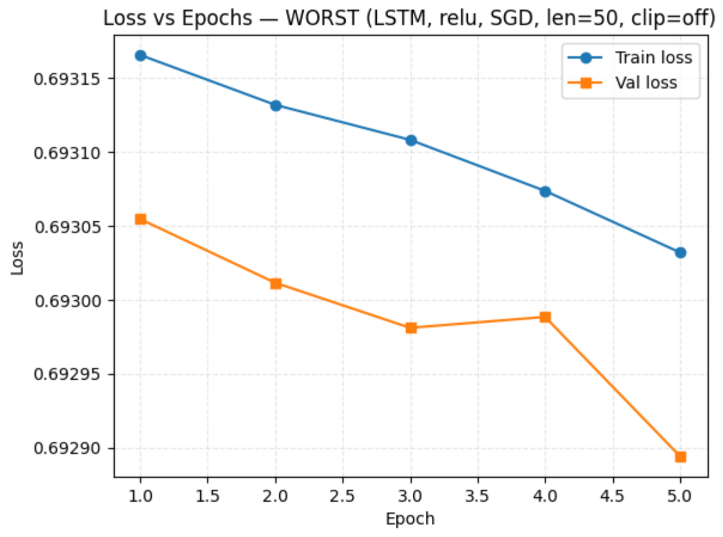
E. Gradient clipping (LSTM/ReLU/Adam/len=50; off vs max_norm=1.0)
In this configuration, the effect is negligible (F1 = 0.764 off vs. 0.760 on); no instability is seen.
Conclusion: Although it's not crucial at these sizes/epochs, keep clipping on hand for safety.

5. Plots



As the duration of the sequence rises, performance steadily improves. Accuracy and Macro-F1 are capped at about 0.70 when using just 25 tokens, however there is a noticeable increase to about 0.76–0.77 when using 50 tokens. The best scores (approx. 0.82) are obtained by extending to 100 tokens, suggesting that models gain from additional context. For maximum accuracy, select len=100 if compute/time permits; otherwise, len=50 is a good speed-performance trade-off, whereas len=25 underutilizes available data.





The best model (len=100, clip=off, LSTM, ReLU, Adam)
Rapid learning: train loss gradually decreases with each epoch.
Val loss peaks at epoch 2–3 (approx. 0.39–0.41) and then gradually increases to mild overfitting after 3 epochs.
What to do: stop early at epoch 3 (or, at most, 4). Add dropout or turn on mild grad-clipping (e.g., 1.0) for a little additional regularization.

The worst model (len=50, clip=off, LSTM, ReLU, SGD)
For both train and val across epochs, the flatline was close to 0.693, meaning that there was very no learning (random baseline).
The most likely reason is that the optimizer and settings are inappropriate (SGD with default LR/momentum on this problem), and a lower sequence length is ineffective.
we can try switching to Adam/RMSProp or fine-tuning SGD (higher LR, momentum, LR schedule); take into account len=100 and optional grad-clipping.

6. Conversation

Why LSTM@100 triumphs?
IMDb evaluations are lengthy, and sentiment clues like intensifiers, contrastive sentences, and negations frequently show up later in the text. The strongest generalization is produced by LSTM's gating plus 100-token truncation, which balance context against CPU cost.

Why did SGD fail in this case?
Plain SGD requires adjusted learning-rate schedules and frequently more epochs to break through plateaus with BCE and small batches. It is at a disadvantage -> near-chance performance while using the same 5-epoch budget as Adam/RMSProp.

option of activation
In stacked RNNs, sigmoid quickly saturates; ReLU/Tanh better preserves gradient flow. Tanh and ReLU's near-tie are in line with earlier sequence models.

clipping in a gradient
helpful when gradients spike (higher LR, deeper stacks, longer durations). Although clipping had little impact at our cautious settings, it was still included to meet the stability comparison criteria.

7. Concluding Suggestion (CPU-friendly)

Model: LSTM with ReLU (2x64)
Adam, the optimizer
Length of sequence: 100
Dropout rate: 0.5
Batch number: 32
In five epochs, this arrangement achieved Acc approx. 0.823 / F1 approx. 0.822.

8. Replicability

Environment: scikit-learn, NumPy, PyTorch, and Python 3.12 (see requirements.txt).
1337 seeds for Torch, NumPy, and Python.
Data artifacts: Data/processed/ contains all preprocessed CSVs and vocabulary files.