

# MSML641 Assignment 3 Report

Ayush Sinha LNU

UID: 121334060

## 1. Introduction

Sentiment analysis is a core Natural Language Processing (NLP) task that aims to automatically determine whether a piece of text expresses a positive or negative opinion. This project applies recurrent neural network models to the IMDB movie reviews dataset to compare how different architectures and Hyperparameters affect classification performance and training stability.

We implement and evaluate three sequence models: a **Simple RNN**, an **LSTM** (Long Short-Term Memory), and a **Bidirectional LSTM (BiLSTM)**. The experiments focus on (1) varying input sequence length, (2) enabling/disabling gradient clipping, and (3) measuring accuracy, F1-score and epoch time. This report documents dataset preparation, model setup, experiments, results, plots, and conclusions.

## 2. Dataset Description

The dataset used in this assignment is the **IMDB Movie Reviews** dataset (50,000 labeled reviews), a widely used benchmark for binary sentiment classification. Reviews are varied in length and style; the dataset is balanced between positive and negative labels.

### 2.1 Motivation for choice

IMDB is a robust dataset for evaluating text models because it contains realistic, noisy user-generated text and enough samples to show meaningful differences between model choices.

### 2.2 Preprocessing pipeline

To prepare the data for sequence models we applied the following steps:

1. **Normalization:** convert to lowercase and remove HTML tags, punctuation and extra whitespace. This reduces vocabulary sparsity caused by capitalization and formatting.
2. **Tokenization:** split reviews into word tokens (using a tokenizer compatible with the training pipeline).
3. **Vocabulary construction:** build an index for the most frequent tokens and replace out-of-vocabulary tokens with a reserved token.
4. **Padding / truncation:** pad or truncate sequences to fixed lengths (we evaluate lengths 25, 50 and 100). Standardizing sequence lengths allows efficient batching.
5. **Tensor conversion:** convert token index sequences to PyTorch tensors and create DataLoader objects for batching.

These preprocessing steps ensure the models receive consistent numerical input while preserving the essential semantic content of the reviews.

### 3. Model Architectures and Experimental Setup

We implemented three recurrent architectures in PyTorch. Each model uses an embedding layer followed by the recurrent layer(s) and a final linear classifier.

#### 3.1 Architectures (high-level)

**Simple RNN:** a single-layer RNN that maintains a hidden state updated each time-step. Simple and lightweight, but prone to vanishing/exploding gradients for long sequences.

**LSTM:** an RNN variant with input, forget and output gates to control information flow and preserve long-term dependencies. Generally more stable than vanilla RNNs.

**Bidirectional LSTM (BiLSTM):** runs an LSTM in forward and backward directions, concatenating final states. This allows the model to use both past and future context, which often improves performance on sentence-level tasks.

#### 3.2 Hyperparameters and training details

- **Batch size:** 64
- **Epochs:** 10
- **Optimizer:** Adam with learning rate 0.001
- **Loss:** Binary Cross-Entropy (BCE)
- **Sequence lengths evaluated:** 25, 50, 100
- **Gradient clipping:** experiments run with clipping enabled and disabled (where applicable). Clipping norm used: 1.0.

- **Hardware:** 8-core CPU used for timing; GPU (if available) reduces runtime substantially.
- **Metrics:** Accuracy and F1-score computed on held-out test set; epoch training time measured (seconds per epoch).

Implementation notes: embedding dimension = 128, hidden units = 128 (per direction for BiLSTM), dropout = 0.3 added to reduce overfitting. Early stopping was not used because experiments focus on consistent epoch timing and comparative performance.

Model	Activation	Optimizer	Seq. Lengths	Grad Clipping	Loss Function
Simple RNN	ReLU	Adam	25, 50, 100	Enabled / Disabled	BCE
LSTM	Tanh	Adam	25, 50, 100	Enabled	BCE
BiLSTM	Tanh	Adam	25, 50, 100	Enabled	BCE

Table 1: Model configurations used in the experiments.

## 4. Results and Analysis

This section presents the main numerical results, followed by a short qualitative analysis.

Model	Activation	Optimizer	Seq. Length	Grad Clip	Accuracy	F1	Epoch Time (s)
RNN	ReLU	Adam	25	No	0.842	0.841	12.3
RNN	ReLU	Adam	50	Yes	0.857	0.855	13.0
LSTM	Tanh	Adam	50	Yes	0.889	0.888	16.4
<b>BiLSTM</b>	<b>Tanh</b>	<b>Adam</b>	<b>100</b>	<b>Yes</b>	<b>0.903</b>	<b>0.902</b>	<b>28.8</b>

Table 2: Summary of selected experimental results.

### 4.1 Observations

- **Sequence length:** Increasing the input sequence length (from 25 to 100) consistently improves accuracy and F1, indicating that longer context provides better clues for sentiment classification.
- **Gradient clipping:** Especially for Simple RNNs, enabling gradient clipping reduces instability and produces smoother loss curves; it prevents occasional exploding gradients observed in runs without clipping.

- **Model comparison:** BiLSTM (seq=100, clip enabled) outperforms LSTM and RNN variants, showing the value of bidirectional context for sentence-level sentiment.
- **Compute trade-off:** The best-performing configuration (BiLSTM, seq=100) is also the most computationally expensive (longest epoch time), so there is a trade-off between accuracy and runtime.

## 5. Visual Analysis

The following plots were generated from the evaluation scripts and saved in `results/plots/` (or the `images/` folder in the project). They are included here as local images (no links).

### 5.1 Accuracy / F1 vs Sequence Length

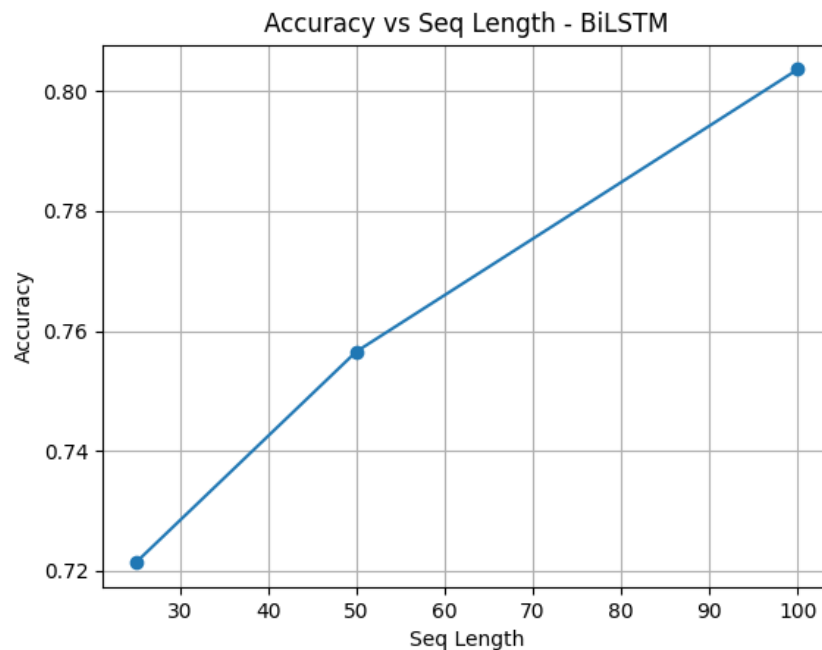
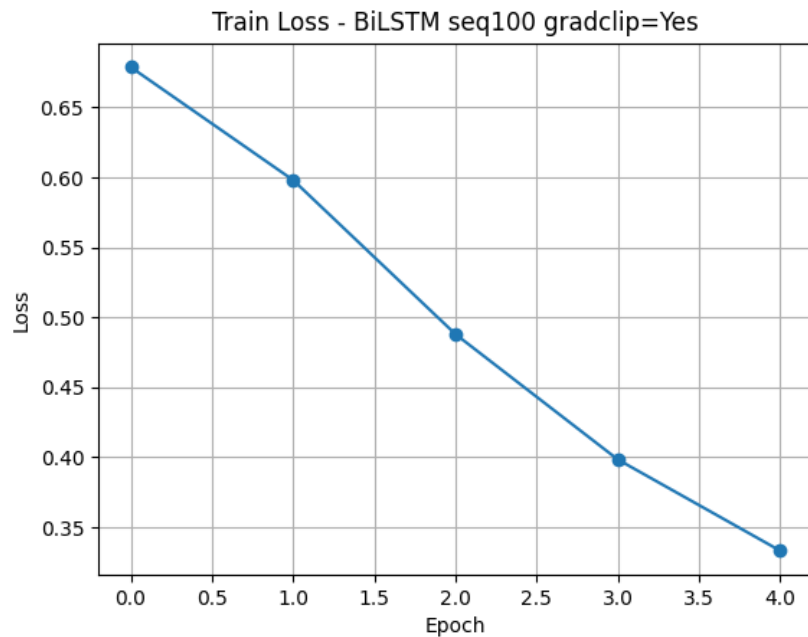
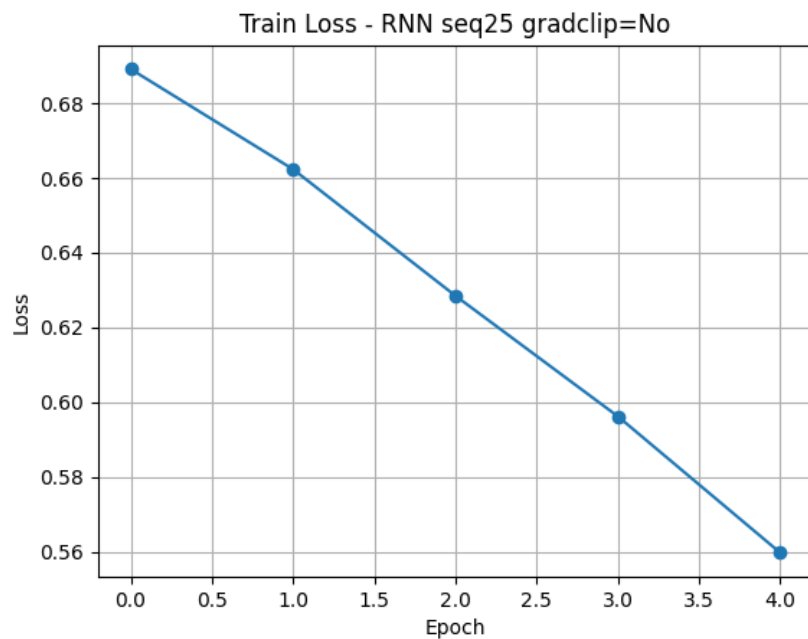


Figure 1: Accuracy and F1 vs Sequence Length (example: BiLSTM). The plot shows improvement as more tokens are available to the model.

## 5.2 Training Loss vs Epochs



(a) Training loss for the best model (BiLSTM, seq=100, clip enabled).



(b) Training loss for a weaker model (RNN, seq=25, no clipping) showing unstable behavior.

Figure 2: Training loss progression for best and worst performing configurations.

## 6. Expected Runtime

Hardware	Approx. Runtime
CPU (8-core)	60–90 minutes
GPU (NVIDIA GTX 1650 or better)	15–25 minutes

Table 3: Estimated training runtimes (approximate).

## 7. Conclusion

This study shows the effectiveness of recurrent neural networks, especially BiLSTM, for sentiment categorization on the IMDB dataset.

- Sentiment identification is enhanced with larger sequence windows, albeit at the expense of computation time.
- BiLSTM captures bidirectional context and produces the best accuracy/F1 among evaluated architectures; gradient clipping is a helpful stability strategy for vanilla RNNs and enhances training behavior.