

CSCE 479/879 Homework 3: Sentiment Analysis with Sequential Models

Derek DeBlieck, Sabrina Fowler, Grace Hecke, Abby Veiman

November 18, 2025

Abstract

This report examines binary sentiment classification on the IMDB movie review dataset using two sequential deep-learning architectures: an LSTM and a Transformer. The goal was to compare how their structural differences influence performance, learning behavior, and overfitting. Both models were trained under identical preprocessing, tokenization, and optimization settings. Results showed that while the LSTM provided a strong baseline, the Transformer achieved slightly higher test accuracy and learned more quickly. It also overfit more sharply, with validation accuracy peaking early. Overall, the experiments show clear trade-offs between recurrent and attention-based models in sentiment analysis.

1 Introduction

The goal of this assignment was to apply two different sequential deep-learning architectures—an LSTM and a Transformer—to the IMDB sentiment analysis dataset and compare their performance using controlled experimental setups. The problem centers on classifying movie reviews as positive or negative, a well-studied task that remains useful for evaluating how model architecture choices influence real-world text-processing performance. To study this, we trained two configurations of each model type and evaluated them using the same preprocessing pipeline and performance metrics.

2 Problem Description

The task for this project was binary sentiment classification on the IMDB movie review dataset containing 50,000 labeled text reviews, with classes evenly split between positive and negative sentiment. The learning problem involved determining whether a given review expresses a positive or negative sentiment. Because movie reviews contain a wide range of writing styles as well as informal language, the dataset provides a practical sample for evaluating the strengths and limitations of sequential deep-learning architectures [1].

This problem is particularly interesting from a modeling standpoint because sentiment analysis requires more than counting individual words. Models need to interpret context, handle instances of negation, and track dependencies across long sequences. Traditional recurrent architectures such as LSTMs were built to manage this type of sequential structure, while modern Transformer-based models have shown strong performance across many language tasks thanks to their ability to model relationships between all token positions at once. By training an LSTM and a Transformer on the same dataset under comparable conditions, we can examine how the differences in architecture affect their training behavior, generalization ability, and susceptibility to overfitting. This comparison helps illustrate how model choice influences results in real-world text classification tasks.

3 Approaches

In order to see how different sequential architectures handle sentiment classification, we implemented two model families: an LSTM with additive attention and a Transformer block.

LSTM with Additive Attention

The first architecture followed a traditional recurrent design but strengthened with an attention mechanism. The model begins with an embedding layer of dimension 128 or 256 (depending on the configuration), followed by a single LSTM layer that returned the full sequence of hidden states. Because the LSTM compresses sequential information over time, we added a **AdditiveAttention** layer to let the model focus selectively on the most relevant timesteps when forming its final representation. The last LSTM hidden state acted as the query, while the full sequence served as keys and values. After attention pooling, the output passed through a dense sigmoid layer for binary sentiment classification.

Both LSTM configurations incorporated L2 regularization on embeddings, recurrent weights, and the output layer, reflecting the need to control overfitting on relatively short training sequences. The configurations differed primarily in scale with the smaller version using 128-dimensional embeddings and 64 LSTM units, and the larger version doubling both the embedding dimension and LSTM width. The second configuration also reduced the learning rate to account for the increased capacity. These variations allowed us to see how model size influences generalization and learning stability.

Transformer Block

The second architecture used a simplified Transformer block centered on multi-head self-attention. After an embedding layer, the sequence passed into a custom **TransformerBlock** consisting of a 4-head **MultiHeadAttention** layer, a feed-forward network, two layer-normalization steps, and dropout. Unlike the LSTM,

which processes tokens sequentially, the Transformer attends to all positions simultaneously, making it more effective at capturing long-range dependencies without relying on an ever-compressed hidden state.

To keep training efficient, this Transformer was intentionally lightweight. The embedding dimension was fixed at 128 in both configurations, and the number of heads was kept at four. The main difference between configurations was the size of the feed-forward network and the dropout rate. After the transformer block, a global average pooling layer reduced the sequence to a fixed-length vector before feeding into a final sigmoid classification layer. This architecture was designed to test whether self-attention alone—not a full multi-layer transformer—could outperform recurrent models on this dataset.

Training Strategy

All models were trained with the Adam optimizer using binary cross-entropy loss. Batches of 64 reviews were used to speed up training, and early stopping with a patience of five epochs prevented unnecessary overtraining while restoring the best validation checkpoint. By keeping the data pipeline and training procedure identical across models, the comparison isolates the effect of architectural differences rather than preprocessing or optimization factors.

4 Experimental Setup

All experiments were carried out using TensorFlow and TensorFlow Datasets with the IMDB movie reviews dataset. To ensure proper separation for model selection, the training portion was further divided into a 90% training set and a 10% validation set.

Before training, all text was processed with a Keras `TextVectorization` layer adapted only on the training split to avoid data leakage. The vocabulary was limited to the 10,000 most frequent words in the training corpus, and each review was converted into a sequence of integer token IDs. All sequences were padded or truncated to a fixed length of 128 tokens to allow efficient batching and ensure consistent input size across models. After vectorization, the training, validation, and test datasets were batched with a batch size of 64 and configured with prefetching for improved throughput.

Each model was trained for a maximum of fifteen epochs, although early stopping often terminated training sooner. Early stopping monitored validation loss and restored the best-performing weights after five consecutive epochs without improvement. Models were trained using binary cross-entropy loss and optimized with the Adam optimizer. Accuracy was the primary evaluation metric during training, validation, and testing.

4.1 Model Hyperparameters

Hyperparameters varied across the four models, primarily in embedding dimension, the size of recurrent or attention components, learning rate, and regularization. Table 1 summarizes the key differences between configurations.

Table 1: Hyperparameter settings for LSTM and Transformer models

Model	Embedding Dim	Units / FF Dim	Dropout / L2	Learning Rate
LSTM Config 1	128	64 LSTM units	L2 = 0.0001	0.001
LSTM Config 2	256	128 LSTM units	L2 = 0.001	0.0005
Transformer Config 1	128	128 FF	Dropout = 0.3	0.001
Transformer Config 2	128	256 FF	Dropout = 0.2	0.0005

Training and validation accuracy were recorded at each epoch to analyze learning behavior, convergence speed, and the onset of overfitting. After training, each model was evaluated on the held-out test set to determine final classification accuracy. Using consistent preprocessing, data splits, and evaluation procedures ensured that observed differences in performance were attributable to architectural variations rather than inconsistencies in training or data handling.

5 Experimental Results

Our first set of observations concerns the test-set accuracy achieved by each configuration. As shown in Figure 1, the LSTM models achieved accuracies of 0.8192 and 0.8078, while the Transformer configurations performed slightly better, reaching 0.8253 and 0.8316. This difference, while not dramatic, reflects the Transformer’s strength in capturing long-range dependencies in text compared to recurrent models. [1]

Training behavior also differed across architectures. Figure 2a shows that the Transformer models learned faster and reached higher training accuracy than both LSTM configurations, with Transformer Config 1 exceeding 95% accuracy by the fifth epoch. However, the validation curves in Figure 2b reveal a more nuanced story: both architectures experienced early peaks followed by mild declines, indicating overfitting after the first few epochs. In particular, Transformer Config 1 showed the sharpest drop in validation accuracy despite its strong training performance.

These observations show that while Transformers provide stronger raw accuracy in this task, they are also more susceptible to overfitting without additional regularization.[1] Overall, both architectures produced competitive results, with the best Transformer model achieving the highest test accuracy in our experiments.

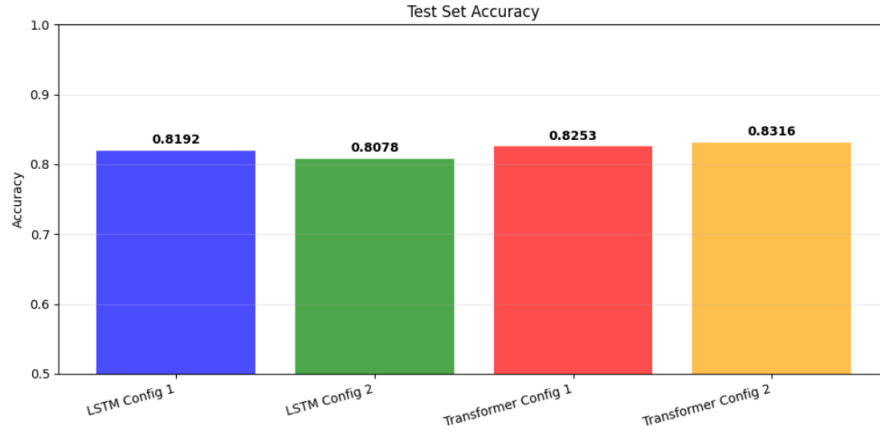
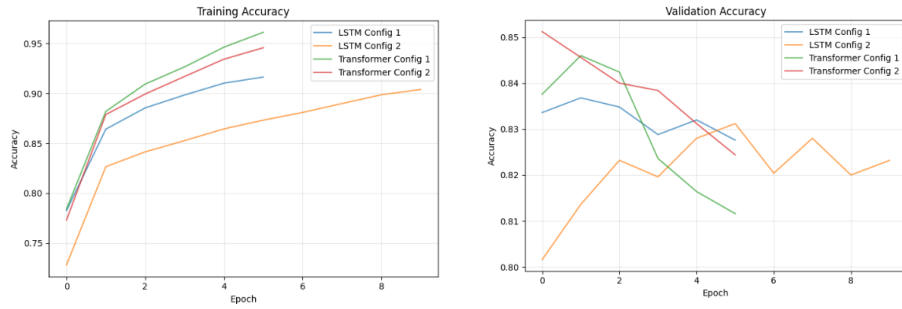


Figure 1: Test-set accuracy for all model configurations.



(a) Training accuracy over epochs.

(b) Validation accuracy over epochs.

Figure 2: Training and validation accuracy for all model configurations.

6 Discussion

The experimental results show clear differences in how the LSTM and Transformer models learned and generalized on the IMDB sentiment classification task. Across all configurations, the Transformer achieved slightly higher peak validation and test accuracy than the LSTM, reflecting its stronger ability to capture long-range dependencies in text. Training curves showed that the Transformer converged more quickly, reaching high accuracy within the first few epochs, while the LSTM was slower it did have steadier improvement. However, the Transformer also displayed more prominent overfitting. Validation loss began to increase even as training loss continued to decrease, and accuracy peaked earlier before gradually declining. The LSTM’s validation metrics remained more stable throughout training, suggesting better regularization under the chosen hyperparameters. These results collectively demonstrate the practical tradeoffs between speed, accuracy, and overfitting when selecting between recurrent and attention based architectures for sentiment analysis.

7 Conclusions

The experiments show that both LSTM and Transformer models are effective for binary sentiment classification on the IMDB dataset, but they exhibit different strengths. Our Transformer model reached higher test accuracy and learned more quickly, demonstrating strong capacity for capturing long-range dependencies in text. Our LSTM, while slightly lower in accuracy, maintained steadier validation performance and experienced less sharp overfitting. The comparison across hyperparameter settings also revealed that model size and regularization influence both convergence speed and generalization. These findings highlight that the choice of sequential architecture directly affects learning dynamics, final accuracy, and robustness, and should align with the priorities of the specific task.

Table 2: Contributions by team member for this assignment.

Team Member	Contribution
Derek & Sabrina	Code Creation
Grace & Abby	Write up

A First Appendix

References

- [1] Shivaji Alaparthi and Manit Mishra. Bidirectional encoder representations from transformers (bert): A sentiment analysis odyssey. *arXiv preprint*

arXiv:2007.01127, 2020. URL: <https://arxiv.org/abs/2007.01127>, doi:
10.48550/arXiv.2007.01127.