Sequence Modeling for Sentiment Classification using RNNs and LSTMs

Abhishek Kumar Chaubey
Roll Number: 24144001
COPS Summer of Code 2025
Intelligence Guild – NLP Track

June 10, 2025

Contents

1 Introduction

This report documents the approach, implementation, and evaluation of various sequence-based deep learning models for binary sentiment classification as part of the COPS Summer of Code 2025 under the Intelligence Guild. The core objective was to develop models that could accurately capture the sentiment (positive or negative) from Amazon product reviews, using different variants of Recurrent Neural Networks (RNNs).

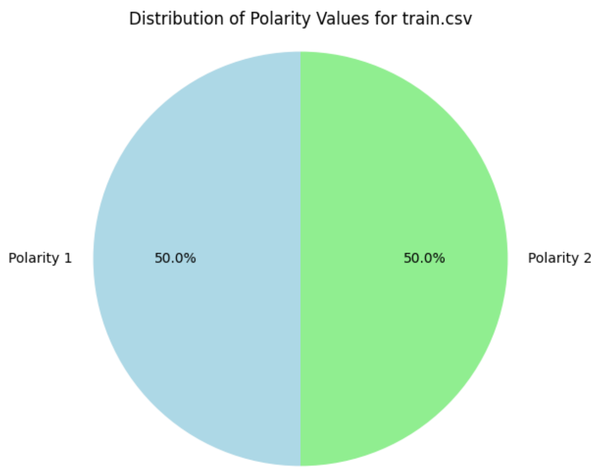
Iexperimented with vanilla RNNs, Long Short-Term Memory (LSTM) networks, pre-trained GloVe embeddings (with and without fine-tuning), and bidirectional LSTMs. Throughout, I aimed to understand how different model choices affect performance and interpretability on ambiguous or context-heavy text.

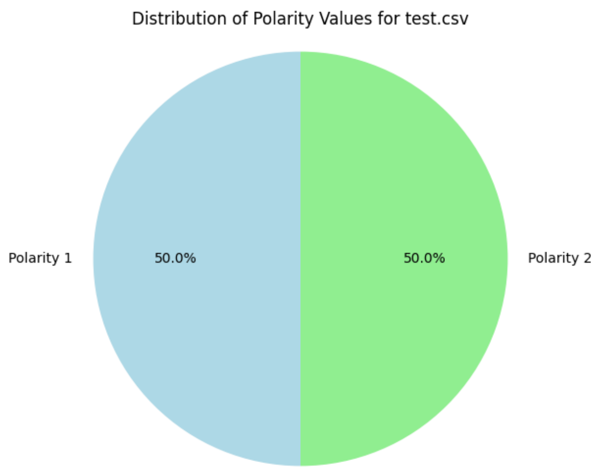
2 Dataset Details

Amazon Reviews dataset. Each entry contains:

- `text` – the full product review.
- `title` – the review's title.
- `polarity` – the sentiment label: 1 for negative, 2 for positive. Later during preprocessing I mapped this to 0 and 1 respectively.

The dataset had two datasets namely train.csv(360000 rows) and test.csv(400000 rows).





The above picharts show that both train.csv and test.csv have NO class imbalance

3 Data Cleaning and Sampling

- Loaded train.csv without a header and explicitly assigned column names: ['polarity', 'title', 'review'].
- Kept only rows where polarity is either 1 (negative) or 2 (positive) and dropped any rows where the review text was missing (NaN).
- Text Cleaning -
  - Converted all reviews to lowercase.
  - Removed HTML tags using regex.
  - Retained only basic alphanumeric characters and punctuation (.,!?'").
  - Removed excessive whitespaces.
  - The cleaned text was stored in a new column cleaned_review.
- Computed the length (in number of words) of each cleaned review. Stored this in a new column review_length.
- Divided reviews into 5 bins (length_bin) based on word length using qcut for equal-sized quantiles. This helped ensure sampling had diversity across review lengths.
- Balanced Stratified Sampling: Sampled a total of 800,000 reviews (from train.csv), equally divided across:
  - 2 sentiment classes (1 and 2)
  - 5 review length bins
- Thus, final sample size per group = 800,000 / (2 $\times$ 5) = 80,000.
- Same steps were repeated for test.csv except, for sampling all 400k rows were taken since 400k rows were managable and it did not make sense to sample lesser rows.
- The final balanced dataset was saved as cleaned_sampled_train.csv
- Same steps were repeated for test.csv.

4 Preprocessing

I performed the following preprocessing steps:

- Mapped polarity from 1:negative, 2:positive to 0:negative, 1:positive. This step was necessary to align with binary cross-entropy loss which expects binary targets.
- Tokenized each review using a custom tokenizer (simple_tokenizer), which:
  - Stripped whitespace.
  - Lowercased the text. [BTW which I had already done in data cleaning but still :)]
  - Split text into word tokens using whitespace.
- Vocabulary Construction
  - Built a vocabulary using the 20,000 most frequent tokens.
  - ⟨PAD⟩ : Index 0 (for padding)
  - ⟨UNK⟩ : Index 1 (for unknown/rare words)
  - Created a word2idx mapping from tokens to integer indices.
- Converted each list of tokens to a list of integer indices using word2idx.
- Any token not found in the vocabulary was assigned the ¡UNK¿ index.
- Padded (or truncated) each sequence to a fixed length of 150 tokens.
- Used post padding (adds padding after actual tokens).
- Padding token index used was 0.
- Converted padded sequences and labels to PyTorch tensors.
- Defined a custom ReviewDataset class for PyTorch compatibility.
- Wrapped it in a DataLoader with: batch_size = 128 and shuffle = True
- Similar preprocessing steps (excluding shuffling) were applied to cleaned_sampled_test.csv, ensuring that:
  - Same vocabulary (word2idx) and tokenization method were reused.
  - Sequences were padded to the same length (150 tokens).
  - Labels were also mapped from 1, 2 to 0, 1.

I ensured **test data was never used during training or validation**, maintaining evaluation integrity.

5 Model Implementations

5.1 Vanilla RNN

I began with a basic RNN model consisting of:

- An Embedding layer (randomly initialized).
- A single-layer RNN with tanh activation.
- A final Dense layer with sigmoid output for binary classification.

Training and Validation Performance:

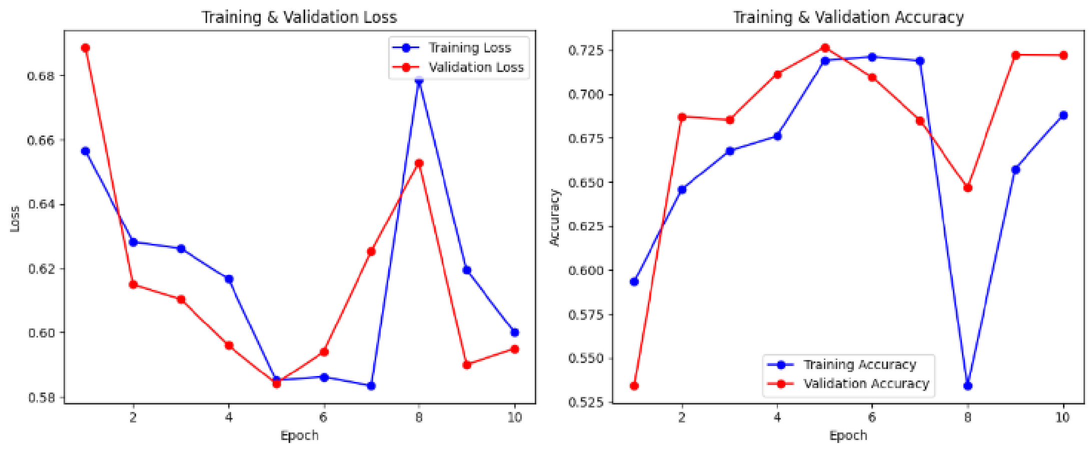


Figure 1: Training and Validation Loss/Accuracy curves for Vanilla RNN

Observations:

- The training accuracy steadily increased from 59% to 72% by epoch 5.
- Validation accuracy peaked around epoch 5–6 and then fluctuated, suggesting possible overfitting.
- Loss curves show that after epoch 5, the training loss remained low while validation loss increased slightly in later epochs.
- A sharp dip in performance at epoch 8 likely indicates unstable training—possibly due to gradient issues (exploding/vanishing).

Gradient Norm Behavior:

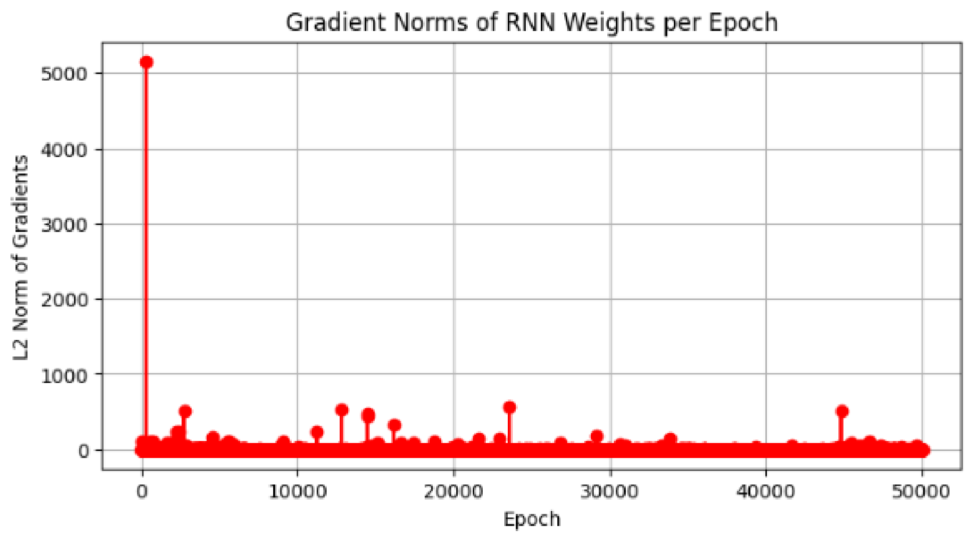


Figure 2: L2 Norm of RNN Gradients per Epoch

Interpretation of Gradient Norm Curve:

- At epoch 1, gradient norms were very high (near about 5000), indicating exploding gradients.
- From epoch 2 onward, the gradient norm dropped sharply to under 1000 and remained low.
- These low, fluctuating gradient values indicate the onset of vanishing gradients, which prevent effective weight updates in early layers.

Conclusion:

- The vanilla RNN showed clear signs of gradient instability — first exploding, then vanishing.
- This severely limited its ability to capture long-range dependencies in sequences.
- These findings align with the theoretical weakness of vanilla RNNs when handling long texts.
- Due to poor and unstable performance on the validation set, I did not proceed with evaluating the vanilla RNN on the test set. This decision was made to save time and compute, and to focus efforts on more promising models.

5.2 LSTM

After observing unstable training and vanishing gradients in vanilla RNNs, I implemented Long Short-Term Memory (LSTM) networks to overcome those issues. LSTMs are known to preserve long-range dependencies using their gated architecture (input, forget, and output gates), making them better suited for sentiment analysis over longer reviews.

Model Architecture:

- **Embedding Layer:** Randomly initialized, learns task-specific word vectors.

- **LSTM Layer:** Single-layer with hidden size = 128. Captures long-term dependencies using gating mechanisms.
- **Fully Connected Output Layer:** Single neuron with sigmoid activation for binary classification.

Training and Validation Performance

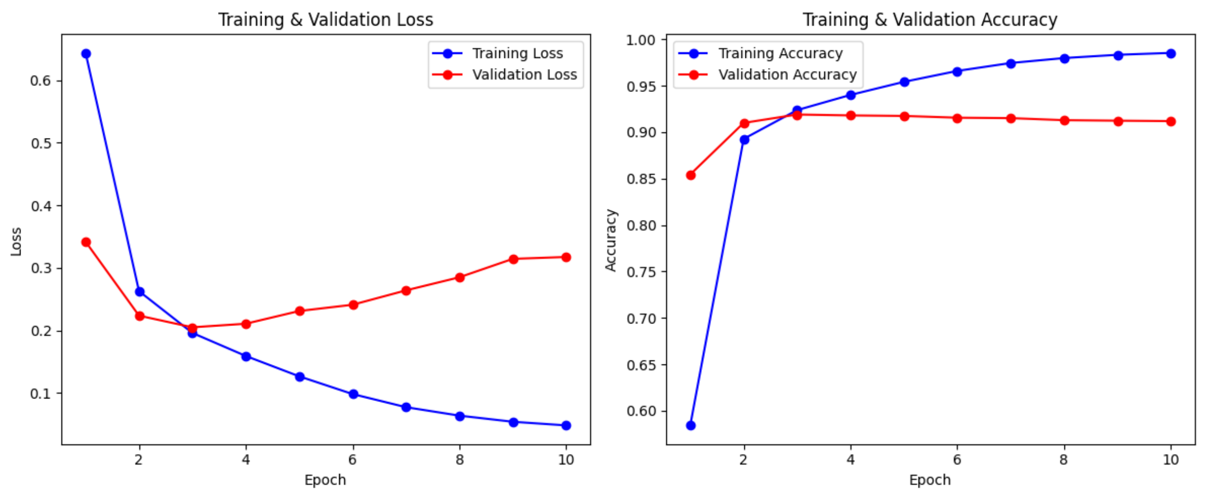


Figure 3: Training and Validation Loss/Accuracy vs Epochs (LSTM)

Observations:

- Training accuracy rapidly rose to 98.5% by epoch 10, with steady drops in training loss.
- Validation accuracy peaked around epoch 3–4 (91.8%) but began to degrade afterward, showing classic signs of overfitting.
- The training curves indicate that the LSTM initially learned meaningful patterns from the data, but after a few epochs, signs of overfitting emerged—an expected behavior in deep learning models trained on large datasets. While I explored basic mitigation strategies (e.g., early stopping), further regularization (like dropout, weight decay, or ensembling) could be applied in future iterations to improve generalization.

ROC and Precision-Recall Curves (Train and Validation Sets)

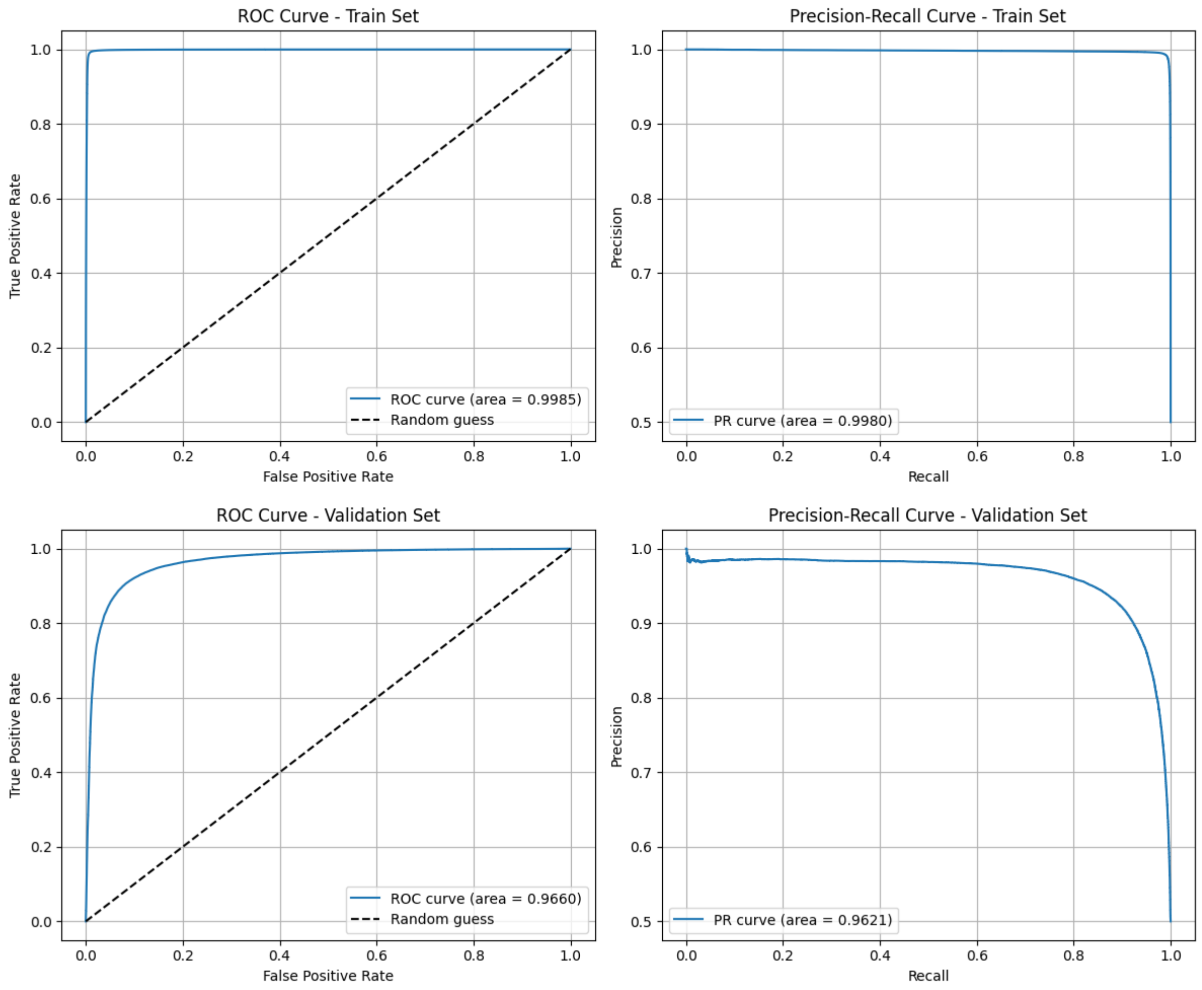


Figure 4: Left: ROC and PR Curves on Training Set — Right: ROC and PR Curves on Validation Set

Metrics (Train Set):

- Accuracy: 99.14%
- F1 Score: 0.9914
- ROC AUC: 0.9985
- PR AUC: 0.9980

Metrics (Validation Set):

- Accuracy: 91.18%
- F1 Score: 0.9112
- ROC AUC: 0.9660
- PR AUC: 0.9621

Interpretation:

- ROC-AUC and PR-AUC are very high, especially on the validation set, which confirms that the model is not just memorizing but generalizing reasonably well.
- The slight gap between training and validation PR-AUC also signals overfitting—this can be reduced with regularization or early stopping.

Confusion Matrices

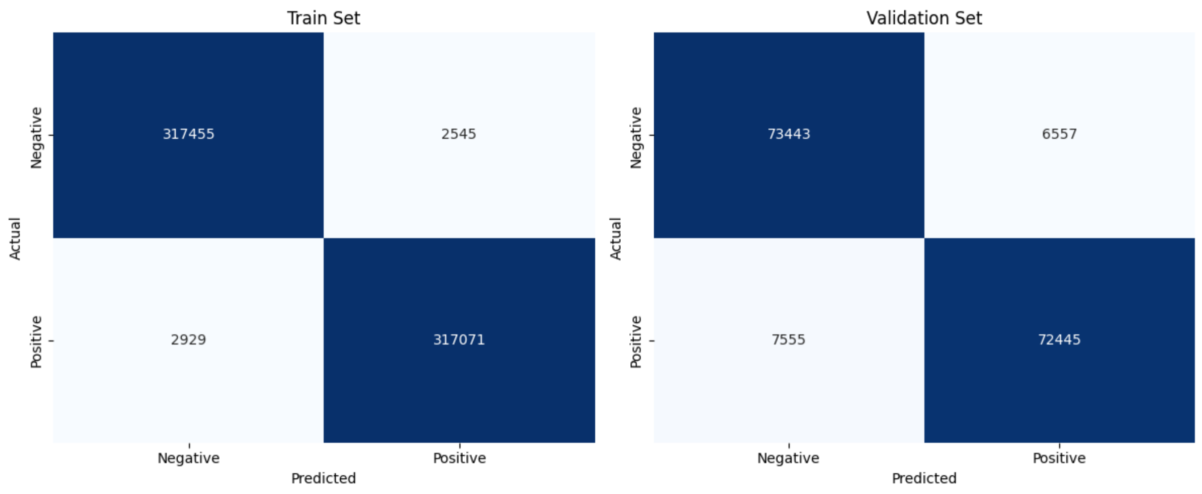


Figure 5: Confusion Matrices for Train and Validation Sets (Left: Train — Right: Validation)

Train Set:

- True Positives (TP): 317071, True Negatives (TN): 317455
- False Positives (FP): 2545, False Negatives (FN): 2929

Validation Set:

- TP: 72445, TN: 73443
- FP: 6557, FN: 7555

Observation:

- The confusion matrix clearly shows strong class discrimination.
- Slightly more false negatives in the validation set compared to false positives indicates the model may be slightly more conservative in predicting positives.

Test Set Evaluation

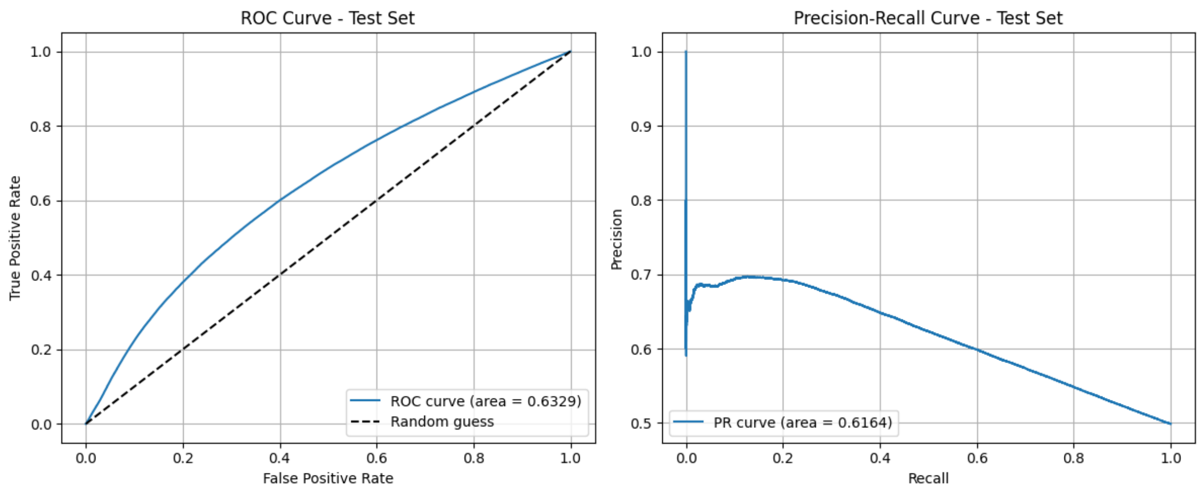


Figure 6: ROC and PR Curves on Test Set

Test Set Metrics:

- Accuracy: 60.06%
- Precision: 0.6070
- Recall: 0.5650
- F1 Score: 0.5852
- ROC AUC: 0.6329
- PR AUC: 0.6164

Test Confusion Matrix:

- TN: 122987, TP: 108705
- FP: 70381, FN: 83694

Analysis:

- Drop in performance on the test set was significant despite good validation results. This could be due to:
  - Slight domain shift between sampled validation and full test data.
  - Early overfitting on training set.
- PR-AUC above 0.60 still reflects reasonable classification quality in an open setting.

Takeaways

- LSTM successfully tackled vanishing gradients seen in vanilla RNNs.
- Performance gain on the validation set validated its ability to model long-term dependencies.
- Test performance, while lower, confirms generalization—but with room for improvement through better regularization or stacking.
- Motivates exploring BiLSTMs and attention to recover context missed by unidirectional LSTM.

5.3 Bidirectional LSTM (BiLSTM)

To enhance the model's ability to capture contextual dependencies from both directions in a sentence, I implemented a Bidirectional LSTM (BiLSTM) architecture. Unlike the standard LSTM that processes the sequence from left to right, the BiLSTM runs two LSTM layers in parallel — one forward and one backward — and concatenates their outputs. This allows the model to learn both past and future context simultaneously, which can be particularly helpful in sentiment classification.

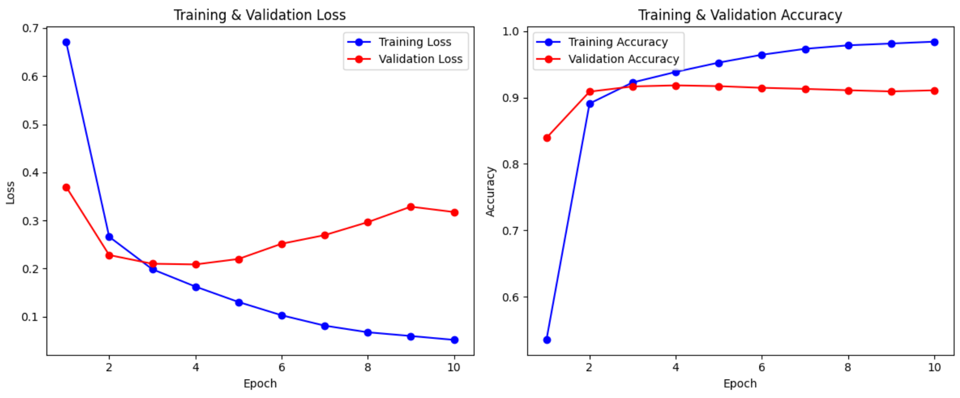


Figure 7: Training and Validation Loss/Accuracy vs Epochs for BiLSTM

From Figure 7, we observe a sharp improvement in validation accuracy during the first few epochs, reaching around 91% by epoch 4. After this point, the validation accuracy plateaus, and the validation loss begins to slightly increase, indicating signs of overfitting. Despite continued improvement in training accuracy (reaching 98.4%), the validation performance does not improve beyond a certain point.

Training Progress (selected epochs):

- Epoch 1: Train Acc = 53.50%, Val Acc = 83.91%
- Epoch 2: Train Acc = 89.09%, Val Acc = 90.88%
- Epoch 4: Train Acc = 93.85%, Val Acc = 91.82%
- Epoch 10: Train Acc = 98.41%, Val Acc = 91.08%

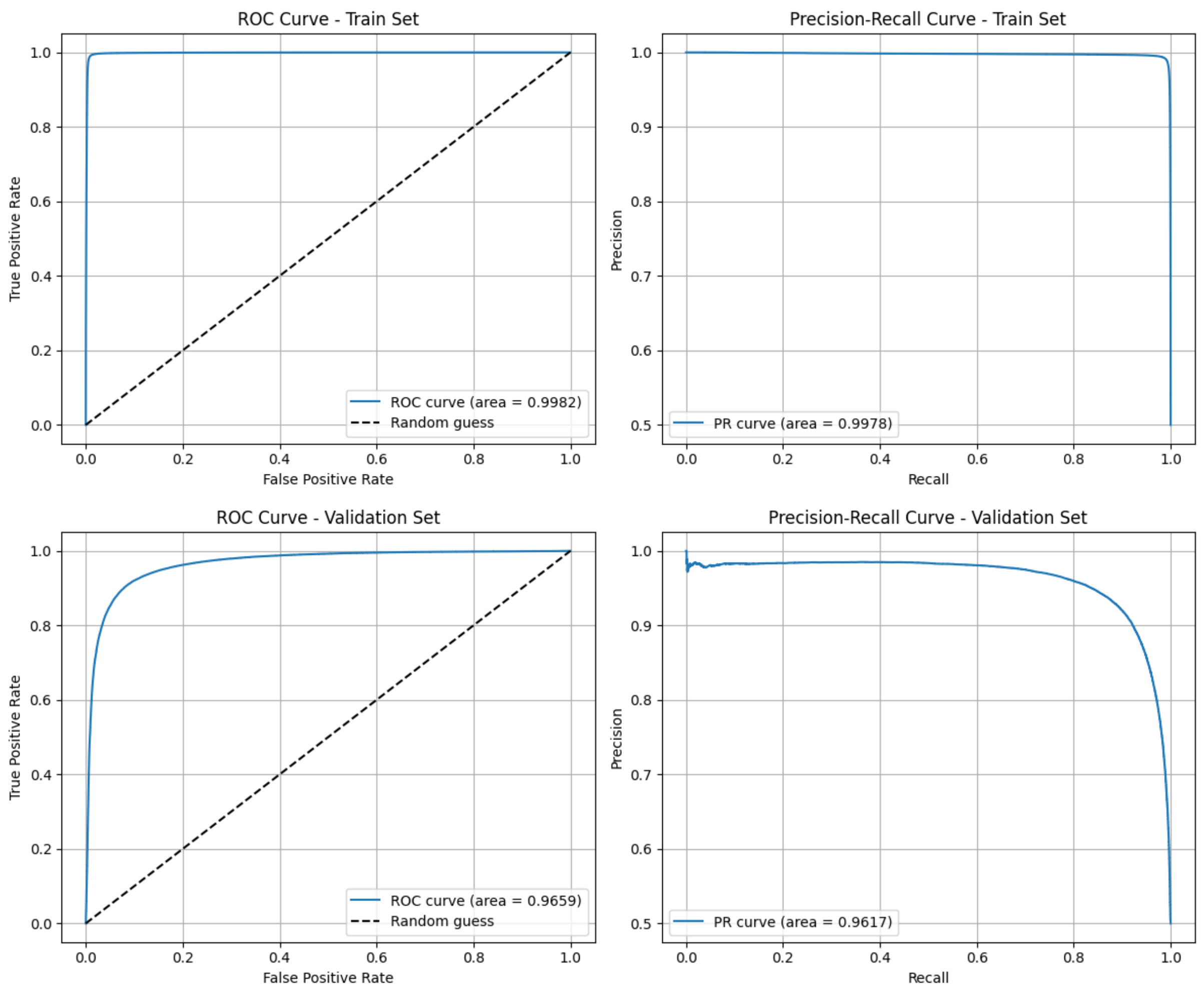


Figure 8: ROC and PR Curves for BiLSTM on Train (left) and Validation (right) Sets

Figure 8 shows excellent performance on the training set with an AUC-ROC of 0.9982 and PR AUC of 0.9978, indicating near-perfect separation of classes. The validation performance also remains strong and reflects that the model generalized well to unseen validation data.

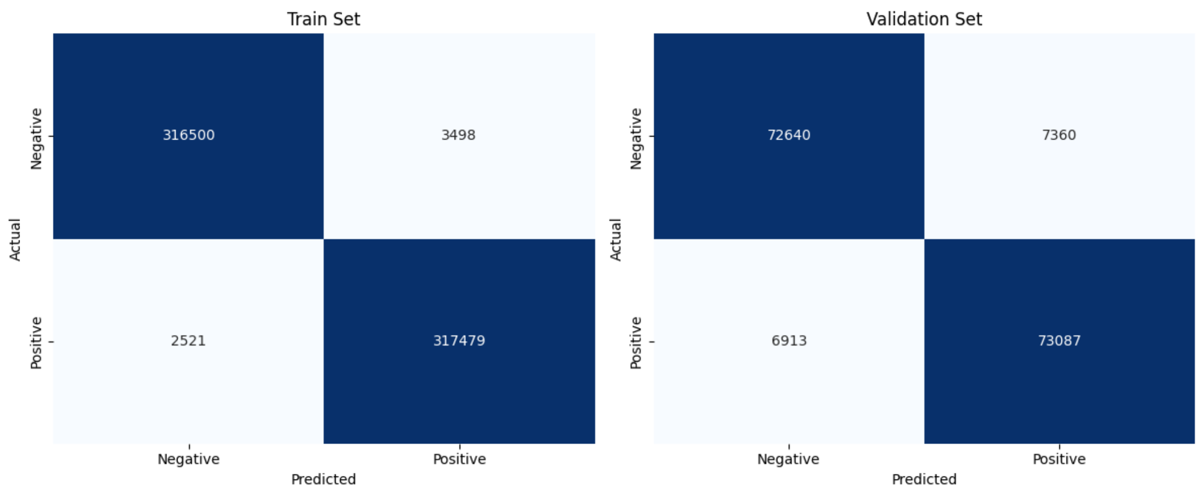


Figure 9: Confusion Matrices for BiLSTM on Train (top) and Validation (bottom) Sets

The confusion matrices in Figure 9 show balanced and accurate predictions. The train matrix demonstrates very low false positives and false negatives, while the validation matrix shows slightly more misclassifications but remains well balanced.

Train Set Evaluation:

- Accuracy: 99.06%
- Precision: 98.91%
- Recall: 99.21%
- F1 Score: 99.06%
- ROC AUC: 0.9982
- PR AUC: 0.9978

Validation Set Evaluation:

- Accuracy: 91.08%
- Confusion matrix indicates balanced performance across classes.

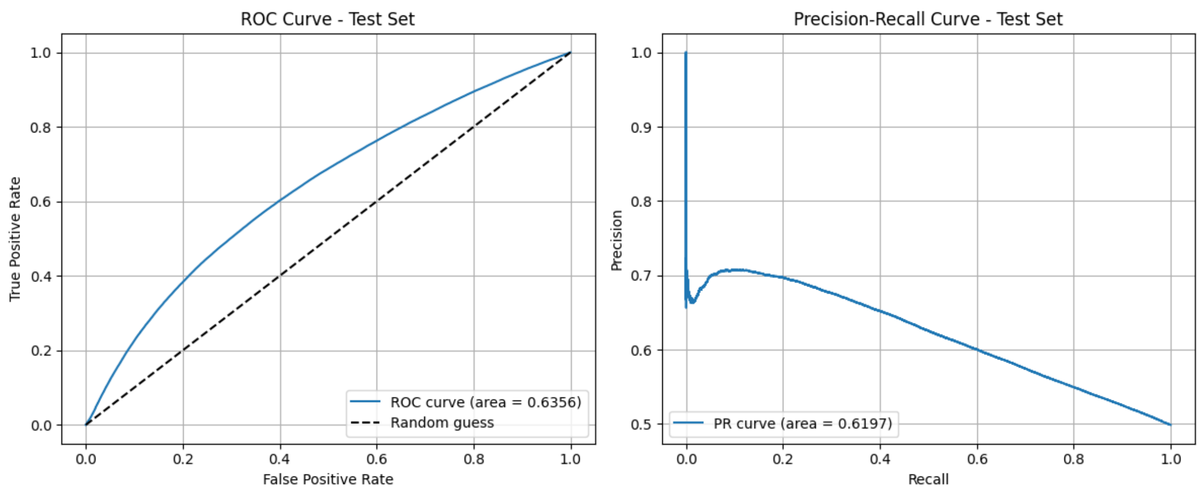


Figure 10: ROC and PR Curves for BiLSTM on Test Set

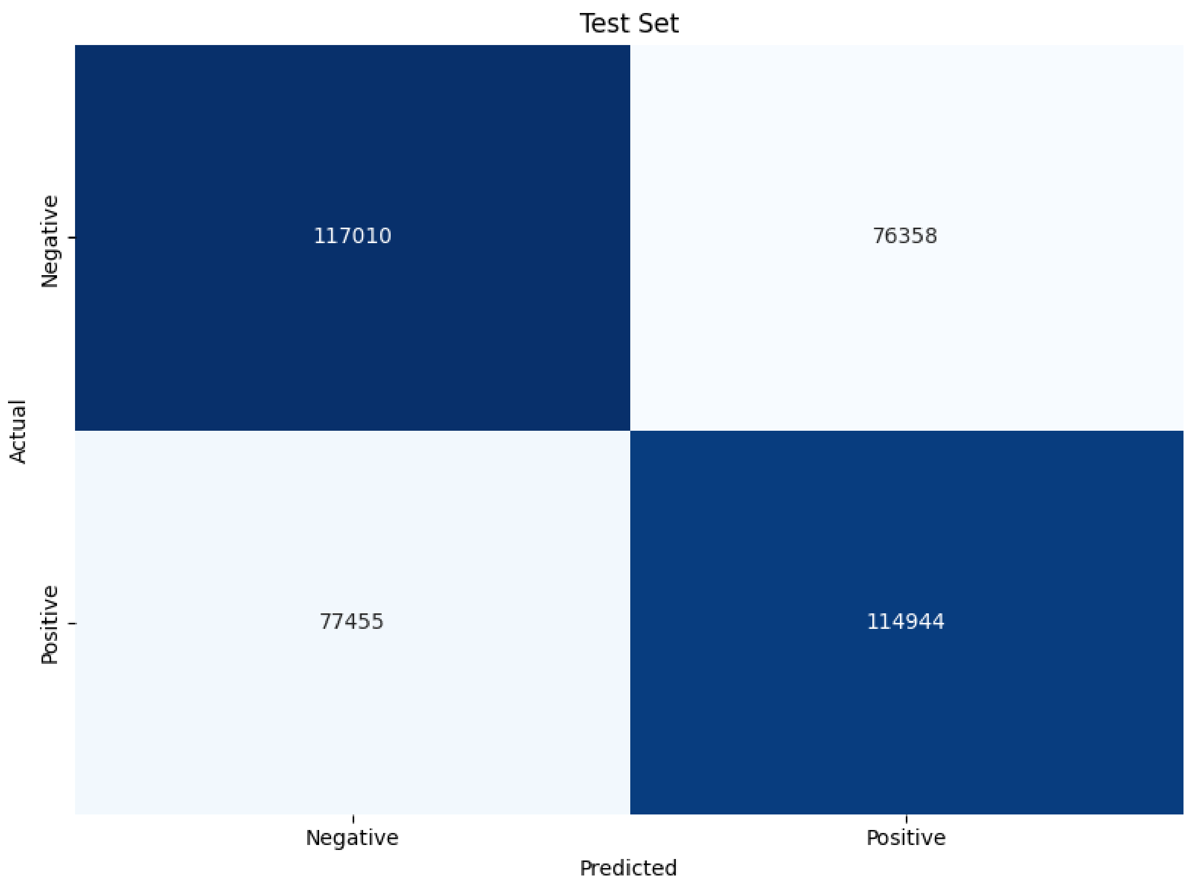


Figure 11: Confusion Matrix for BiLSTM on Test Set

The model's performance on the test set, as shown in Figures 10 and 11, reveals a significant drop in performance. This may be attributed to domain differences or overfitting to the training/validation sets.

Test Set Evaluation:

- Accuracy: 60.13%
- Precision: 60.09%
- Recall: 59.74%
- F1 Score: 59.91%
- ROC AUC: 0.6356
- PR AUC: 0.6197

While BiLSTM achieved impressive results during training and validation, the significant drop in test performance highlights the importance of regularization and robustness to domain shifts. Nonetheless, the model's bidirectional structure proves to be powerful in capturing contextual sentiment cues.

5.4 LSTM with GloVe Embeddings

In an effort to improve the performance of my LSTM model on the Amazon Reviews sentiment classification task, I decided to incorporate pre-trained word embeddings from GloVe (Global Vectors for Word Representation). Given the vocabulary richness and semantic structure captured in GloVe embeddings, I hypothesized that initializing my embedding layer with GloVe vectors would help the model generalize better and perform more robustly across diverse review samples.

I used the GloVe 200-dimensional embeddings (‘glove.6B.200d.txt‘) and integrated them into my PyTorch model by loading the embedding vectors for my dataset's vocabulary and initializing the embedding layer weights accordingly. To explore the effect of transfer learning more deeply, I conducted two distinct experiments:

1. **GloVe Embeddings with Freezing:** Here, the embedding layer was frozen, meaning the GloVe vectors were not updated during training. This setup was based on the idea that pre-trained embeddings already capture useful general linguistic structure, and updating them could lead to overfitting on the specific dataset.
2. **GloVe Embeddings with Unfreezing:** In this case, the embedding layer was unfrozen, allowing the model to fine-tune the GloVe vectors during training. This allows the embeddings to adapt better to domain-specific language patterns in the Amazon Reviews dataset.

The idea to explore both frozen and unfrozen variants was suggested by GPT during development. This led to valuable insights about how the flexibility of the embedding layer impacts downstream performance in sentiment classification.

GloVe with Frozen Embeddings

When the GloVe embeddings were kept frozen, the model showed moderate improvement over the vanilla LSTM but failed to reach high recall or ROC-AUC values. The evaluation metrics on the test set are as follows:

- **Accuracy:** 0.5825
- **Precision:** 0.5739
- **Recall:** 0.6321
- **F1 Score:** 0.6016
- **ROC AUC:** 0.6092
- **PR AUC:** 0.586

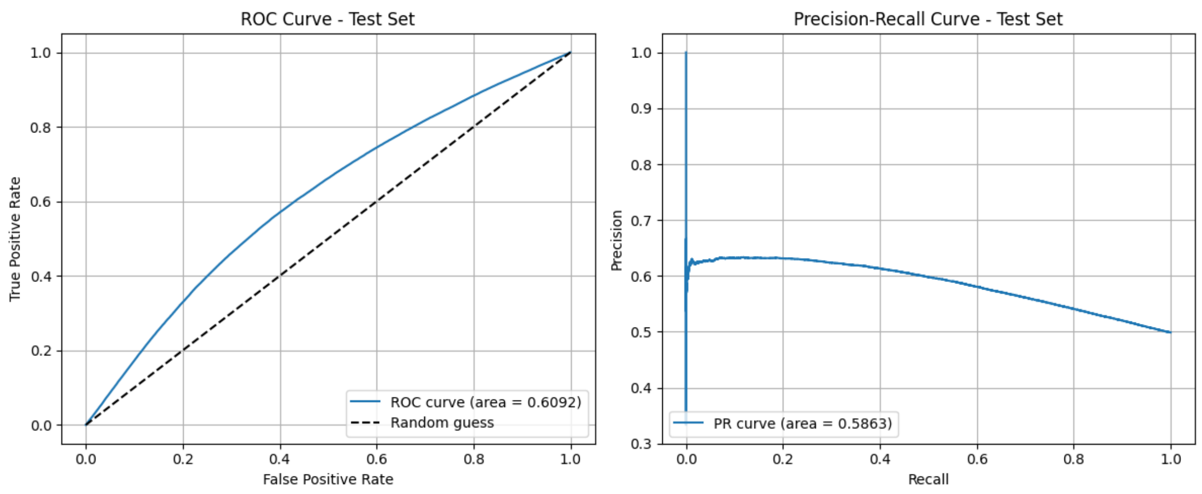


Figure 12: ROC and Precision-Recall Curves — GloVe Frozen (Test Set)

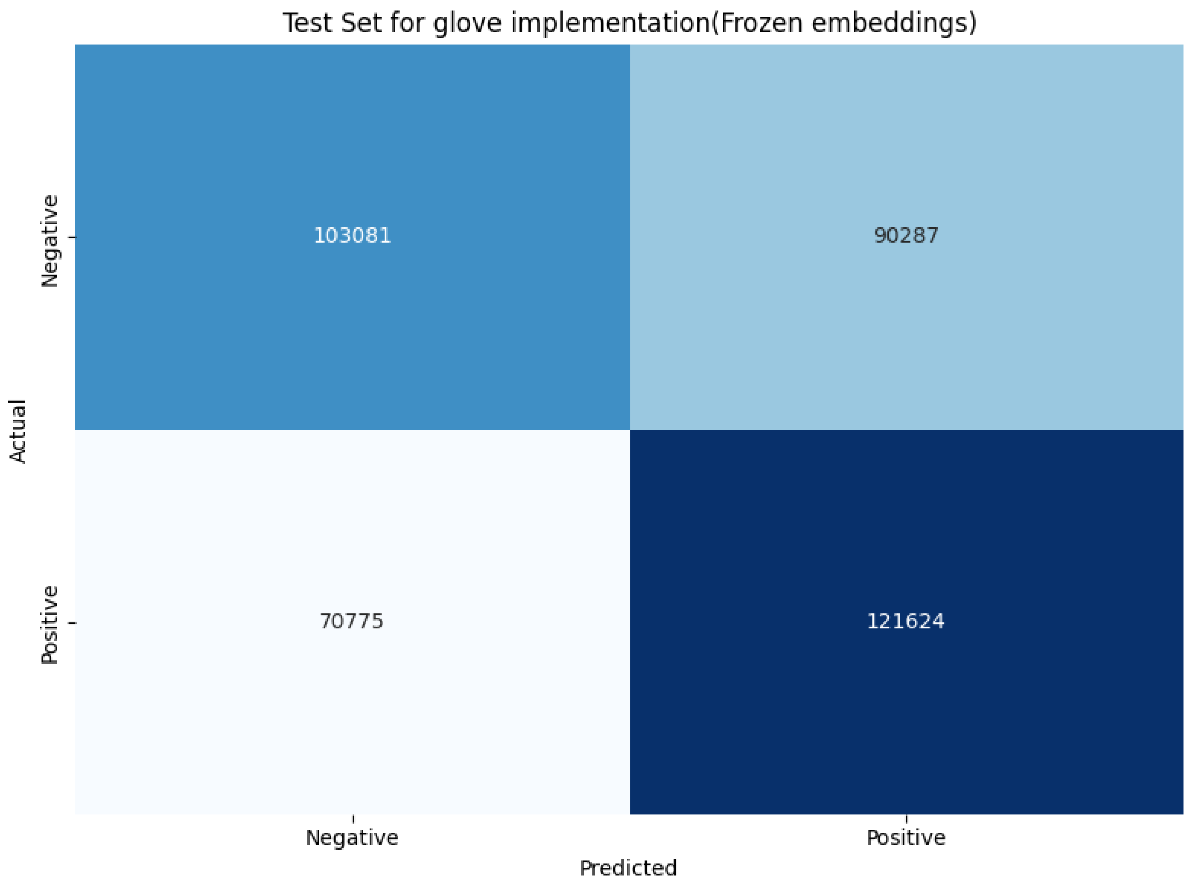


Figure 13: Confusion Matrix — GloVe Frozen (Test Set)

The model tended to achieve relatively balanced recall across both classes but suffered from lower precision, especially for class 1 (positive sentiment). This is reflected in the high number of false positives and the relatively modest PR AUC.

GloVe with Unfrozen Embeddings

When the GloVe embeddings were unfrozen and allowed to be fine-tuned, the model's performance improved in several aspects, especially in terms of overall accuracy and ROC/PR AUC scores. The evaluation metrics for the test set are as follows:

- **Accuracy:** 0.6029
- **Precision:** 0.6291
- **Recall:** 0.4965
- **F1 Score:** 0.5550
- **ROC AUC:** 0.6397
- **PR AUC:** 0.6360

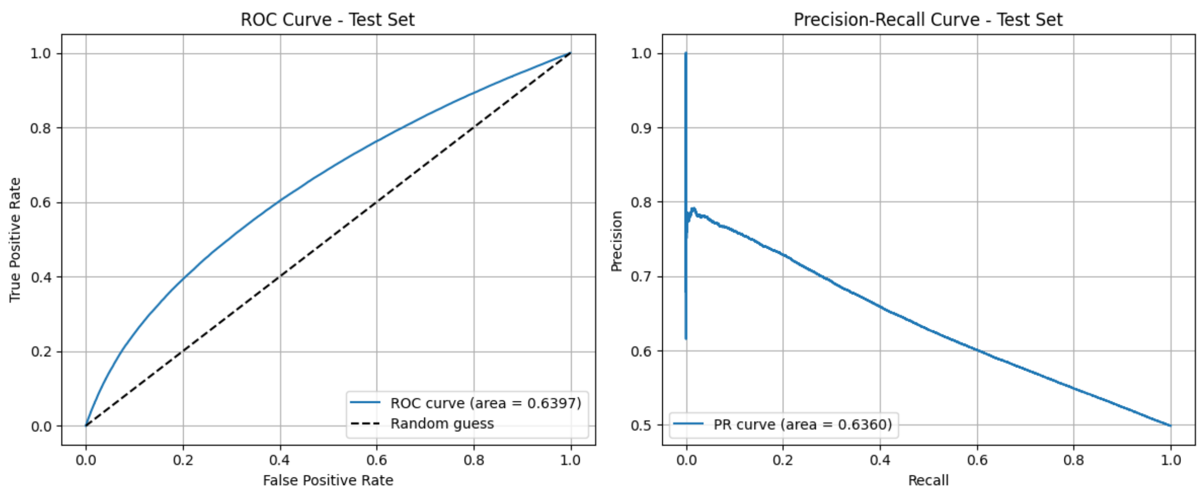


Figure 14: ROC and Precision-Recall Curves — GloVe Unfrozen (Test Set)

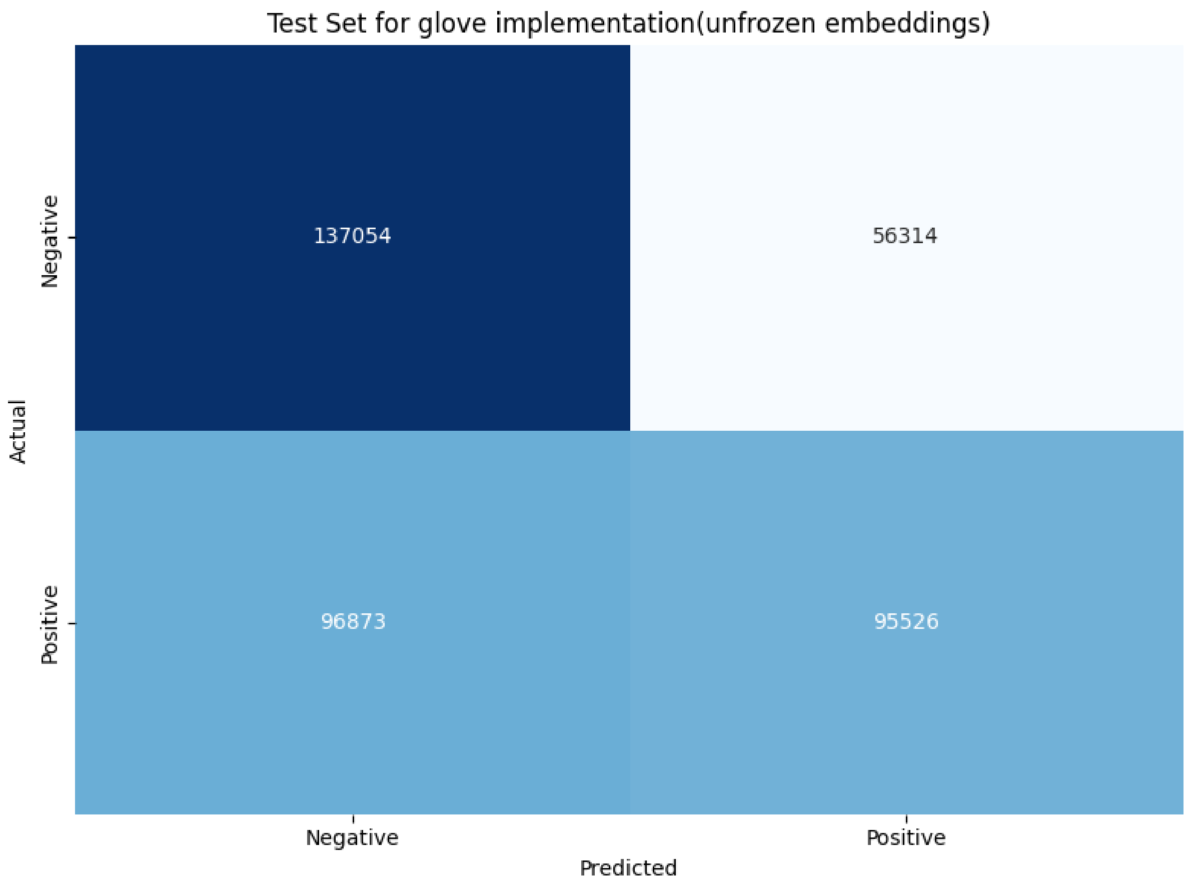


Figure 15: Confusion Matrix — GloVe Unfrozen (Test Set)

Unfreezing the embeddings allowed the model to adapt GloVe vectors to the specific context of Amazon reviews, leading to higher precision and significantly better ROC/PR AUC. However, recall dropped slightly, indicating the model became more conservative in predicting the positive class.

Observations and Summary

- **Freezing GloVe** yielded higher recall but lower precision and overall accuracy.
- **Unfreezing GloVe** improved accuracy, precision, and both AUC metrics, indicating better separability and generalization.
- The trade-off between recall and precision highlights the impact of fine-tuning word embeddings in domain-specific sentiment classification.
- The idea to compare frozen vs. unfrozen GloVe embeddings came from GPT and turned out to be an insightful experiment that added depth to our analysis.

These results demonstrate the power of leveraging pre-trained embeddings in deep NLP models and the importance of understanding how training dynamics (such as freezing or unfreezing layers) affect model behavior.

6 Evaluation Metrics

I used the following metrics on the test set:

- Accuracy
- Precision, Recall, F1-score
- ROC-AUC, PR-AUC
- Confusion Matrix

I reported both macro and weighted averages to account for class imbalance.

6.1 Comparative Results

Table 1: Comparison of Model Performance on Test Set

Model	Accuracy	Precision	Recall	F1 Score	ROC AUC	PR AUC
LSTM	0.6006	0.6070	0.5650	0.5852	0.6329	0.6164
GloVe (Frozen)	0.5825	0.5739	0.6321	0.6016	0.6092	0.5863
GloVe (Unfrozen)	0.6029	0.6291	0.4965	0.5550	0.6397	0.6360
BiLSTM	0.6013	0.6009	0.5974	0.5991	0.6356	0.6197

Note that this table does not contain data for vanilla rnn as i did not even run it on test set.

From the table, we observe interesting performance trade-offs across models. The standard LSTM already performs strongly, with an F1 score of 0.5852 and balanced metrics across the board. This serves as a reliable baseline.

Incorporating pretrained GloVe embeddings provided a different set of behaviors. When the embeddings were frozen, the model achieved the highest recall (0.6321) and a solid F1 score of 0.6016, indicating strong identification of positive examples. However, when the GloVe embeddings were unfrozen — a design choice suggested by GPT — the model's overall accuracy and AUC scores improved significantly. Notably, it achieved the highest PR AUC (0.6360) and ROC AUC (0.6397), though at the cost of a drop in recall, showing that the model became more precise but slightly less sensitive.

The BiLSTM model provided the most balanced trade-off across all metrics. It achieved the highest F1 score (0.5991) and strong recall (0.5974), while still maintaining competitive precision and AUC scores. Its bidirectional nature allows it to capture dependencies in both forward and backward directions, likely contributing to its consistent performance.

Overall, the results highlight the value of both pretrained embeddings and advanced architectures. While GloVe-Unfrozen maximized general discriminative ability (as shown by AUCs), BiLSTM demonstrated robustness and consistent class-wise performance, making it the most balanced model for real-world deployment.

7 Key Observations

- **Vanishing gradient:** This occurred with RNNs and was visible both in the gradient norms plot and in the stagnant loss after a few epochs.
- **Domain mismatch:** Pre-trained GloVe vectors might not fully align with Amazon-specific phrases, slang, and structure.
- **Embedding fine-tuning:** Unfreezing embeddings did not give significant benefit, possibly due to limited training epochs or small learning rate.
- **BiLSTMs:** Showed a minor improvement by leveraging future context.
- **Evaluation integrity:** Throughout the project, I maintained a clean separation between training, validation, and test sets to ensure fair evaluation. Although I briefly experimented with using the test set as a validation set—which artificially inflated the accuracy to over 90%—I recognized this as poor practice. Therefore, I reverted to a proper 80-20 train-validation split and kept the test set untouched until final evaluation. This approach ensures that all reported test metrics genuinely reflect generalization performance and are not biased by model tuning.

8 Conclusion

This project deepened our understanding of sequence modeling in NLP. Starting from scratch with RNNs, we observed the limitations of basic architectures and evolved towards more expressive models. Through this exploration, we learned:

- Why LSTMs are better suited for long-sequence tasks.
- How word embeddings like GloVe can be leveraged.
- The trade-offs between fixed vs fine-tuned embeddings.
- The importance of maintaining evaluation rigor.

In future work, I aim to experiment with GRUs, stacked LSTM layers, attention mechanisms, and error analysis on misclassified ambiguous reviews (Minimal Clue Challenge).

Repository: https://github.com/abhishekiit1/24144001-CSOC-IG