Sequence-Based Sentiment Analysis on Amazon Reviews

Shreshth Vishwakarma
Roll Number: 24095105

June 2025

Objective

The aim of this project is to build a sequence-based model (RNN/LSTM/GRU) to predict sentiment polarity (positive or negative) from product review text. The model is designed to go beyond keyword spotting, learning to capture subtle tone, sarcasm, and emotional cues in text reviews.

Dataset Summary

- **text**: Full review written by a user.
- **title**: Title of the review.
- **polarity**: 1 → Negative, 2 → Positive.
- **Source**: Amazon Reviews Dataset (see assignment for link).

Preprocessing and Tokenization

- Text cleaning: Lowercasing, removing non-alphabetic characters, and extra spaces.
- Tokenization: Splitting text into tokens (words).
- Padding/Truncating: Each review is post padded or truncated to a fixed length (`MAX_LEN = 256`).
- Vocabulary: Top 30,000 most frequent words retained in the vocab due to memory constraints.

Word Embeddings

- Used pre-trained GloVe embeddings (100-dimensional).
- Embedding matrix constructed for the vocabulary; words not found in GloVe were initialized as zero vectors.

- Embedding layer weights set to non trainable to leverage pre-trained semantics.

Model Architecture

RNN Model

- Embedding layer (frozen or set to non trainable as requires_grad=False is set).
- 2-layer bidirectional RNN to capture context from both past and future elements (hidden size 128).
- Fully connected layer for binary output.
- cannot implement L2 regularization in both RNN and LSTM, which i thought to use but cannot due to runtime disconnection due to usage limit of colab GPU.

LSTM Model

- Embedding layer (frozen or set to non trainable).
- 2-layer bidirectional LSTM to capture context from both past and future elements (hidden size 128, dropout 0.5 is applied to avoid overfitting).
- Fully connected layer for binary output.

Training Details

- Loss: Binary Cross-Entropy with Logits, as it use sigmoid itself rather than running sigmoid as a separate layer in the model and then passing its output to BCELoss.
- Optimizer: Adam, learning rate 0.001. Tried with this learning rate but could not tune to other rates otherwise retraining the whole model would be required. But it worked fairly well ig.
- Batch size: 512 used for mini batch GD. I chosed 512 over 32 or 64 as I wanted faster training per epoch.
- Training for 4 epochs due to resource and time constraints.
- Device: CUDA (GPU) Tesla T4 as available on colab.

Evaluation Metrics

- **Accuracy**: Proportion of correct predictions.
- **F1-Score**: Harmonic mean of precision and recall.
- **Confusion Matrix**: Visualizes true positives, true negatives, false positives, and false negatives.

Results

RNN Model

- Test Accuracy: 81.704395%
- Test F1-Score: 0.828627

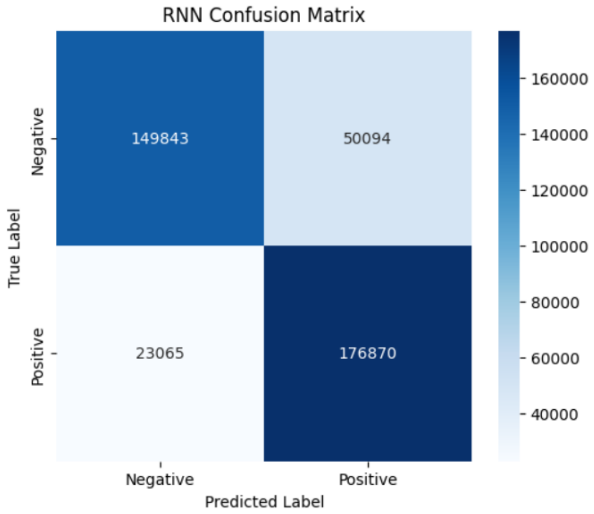

Figure 1: Confusion Matrix for RNN Model

LSTM Model

- Test Accuracy: 95.073674%
- Test F1-Score: 0.951429

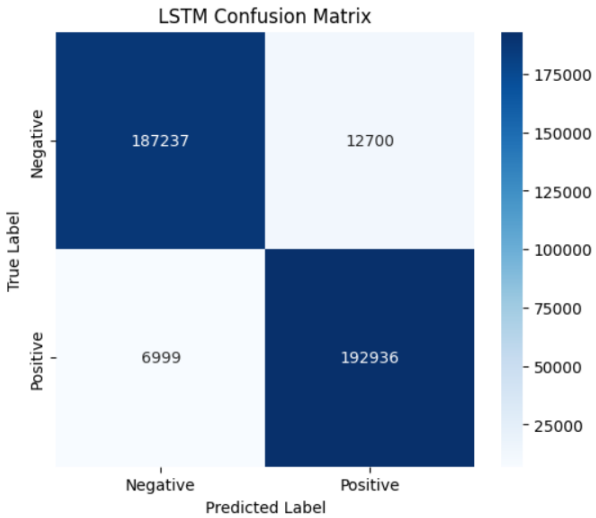

Figure 2: Confusion Matrix for LSTM Model

Error Analysis

- **False Positives/Negatives**: Many misclassifications occur in reviews with ambiguous or sarcastic language.
- **Ambiguity**: Short reviews or those with subtle cues (e.g., 'Great, just great' as sarcasm) are often misclassified.
- **Review Length**: The LSTM model performs better on longer, more complex reviews compared to the RNN due to different complexities used.

Bonus: Minimal Clue Challenge

Examples of ambiguous reviews misclassified by the model:

1. "I bought this and it completely broke after one use. Horrible experience." (Predicted: Positive, True: Negative)

I could have tested on many other text inputs but was only able to test on 5 of it, but the model performed unwell on the questions part.

Analysis:

- **Why misclassified?** The model struggled when the user said he bought it and it thought that if the product is bought then it thought it as a nice review.
- **Human clues:** Humans can detect sarcasm through punctuation, context, or cultural knowledge, which the model lacks.
- **Potential fixes:** I could have used different things like Hyperparameter tuning methods using optuna but was not able to do because runtime disconnected. Incorporate more context (e.g., user history), use transformer based models, or train with explicit sarcasm labels. A more complex model could be incorporated but keeping in mind of overfitting.

Conclusion

- LSTM models outperform vanilla RNNs in capturing long range dependencies and complex sentiments.
- Pre-trained word embeddings significantly improve performance.
- Sarcasm and ambiguity remain challenging. Also improving on the model's hyperparameters will be highly beneficial.

References

Things like integrating with Kaggle api, downloading Glove embeddings directly, making chunks of dataset to incorporate the slow computation due to large dataset size, defining the custom iterable dataset class and analytical part was referenced and taken help from ChatGPT and CampusX videos.