Individual Final Report – Adam Stuhltrager

1. Introduction

This project applied Natural Language Processing (NLP) techniques to analyze Amazon Electronics reviews. The objective was to classify customer sentiment based on review content. The shared work included defining the problem, selecting a dataset, designing a preprocessing pipeline, building and evaluating models, and compiling the final report and presentation. My primary contributions involved developing models used in the project and designing the preprocessing pipeline that underpinned them.

2. Individual Contributions and Explanation

I was responsible for the design and implementation of the LSTM and Baseline models used in the project. I also developed the comprehensive preprocessing pipeline that handled large-scale text data efficiently. My teammates contributed in complementary areas—Ramana added model-saving functionality and developed the CNN model, Chaya developed the RNN model, while both Chaya and Ramana executed training and evaluation routines.

Data Preprocessing

Data Loading and Sampling

- Loaded data from a JSONL file (`Electronics.jsonl`).
- Implemented a sampling mechanism to handle large datasets (default size: 10 million samples).
- Added robust error handling for file loading and JSON parsing.
- Extracted key fields: `rating`, `text`, `title`, `helpful_votes`, and `verified_purchase`.

Text Preprocessing Pipeline

- Converted text to lowercase.
- Removed punctuation, special characters, and numeric values.
- Tokenized using NLTK's `word_tokenize`.
- Removed English stopwords.
- Enabled parallel processing using `joblib` to optimize preprocessing speed.

Sentiment Label Mapping

- Transformed 5-star ratings into sentiment classes:
    - 1–2 stars → Negative (0)
    - 3 stars → Neutral (1)
    - 4–5 stars → Positive (2)

Model Architecture

Baseline Model: Multinomial Naive Bayes

- Utilized TF-IDF vectorization with the following configuration:
    - Maximum features: 20,000
    - N-gram range: (1, 2)
    - Minimum document frequency: 3
    - Maximum document frequency: 0.95
    - Enabled parallel processing
- Trained using Multinomial Naive Bayes with `alpha=0.1` for smoothing.

Advanced Model: LSTM

- Preprocessed text using tokenization and padding.
- Key parameters:
    - Vocabulary size: 20,000
    - Maximum sequence length: 200
- Neural network architecture:
    - Embedding layer (100 dimensions)
    - LSTM layer with 128 units
    - Dropout layer (rate: 0.5)
    - Dense layer with 64 units and ReLU activation
    - Output layer with 3 units and softmax activation

Additional Model: CNN

- Ramana developed and trained a CNN-based text classifier with the following architecture:
    - Embedding layer
    - 1D Convolutional layer
    - Global MaxPooling layer
    - Dense and Dropout layers

    - Final output softmax layer

Training Setup

- Data split: 80% training / 20% test
- Training configuration:
    - Batch size: 128
    - Epochs: 10
    - Early stopping (patience = 3)
    - Learning rate: 0.001
    - Optimizer: Adam
    - Loss function: Categorical Crossentropy
    - Evaluation metric: Accuracy

Performance Optimizations

- Enabled parallel preprocessing for speed.
- Ensured memory-efficient data handling.
- Applied batch processing for large-scale training.
- Supported GPU acceleration (if available).

Evaluation Metrics

- **Accuracy**: Proportion of correct predictions.
- **Precision**: TPTP+FP\frac{TP}{TP + FP}
- **Recall**: TPTP+FN\frac{TP}{TP + FN}
- **F1-Score**: Harmonic mean of precision and recall.

Visualization

- Confusion matrix and training history provided below.

3. Results

Model Performance Comparison (10M Samples)

Model	Accuracy	Precision	Recall	F1-Score
LSTM	0.8916	0.8709	0.8916	0.8741
RNN	0.8901	0.8749	0.8901	0.8799
CNN	0.8866	0.8664	0.8866	0.8713
Naive Bayes	0.8629	0.8303	0.8629	0.8354

Table 1: Performance Metrics for all Models

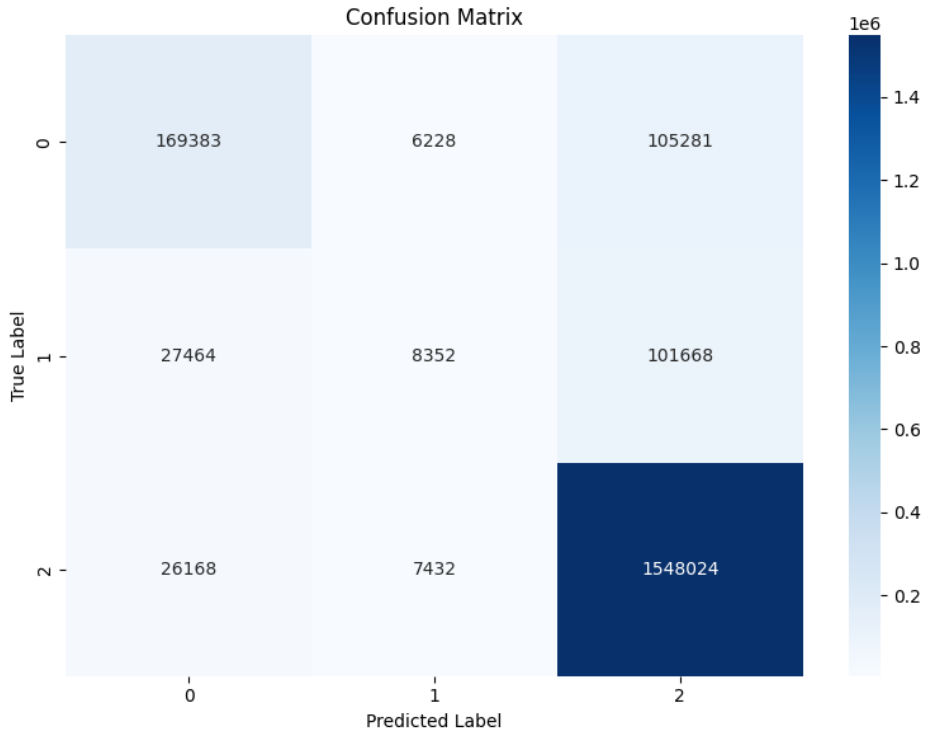

Figure 1: Confusion Matrix for Baseline Model

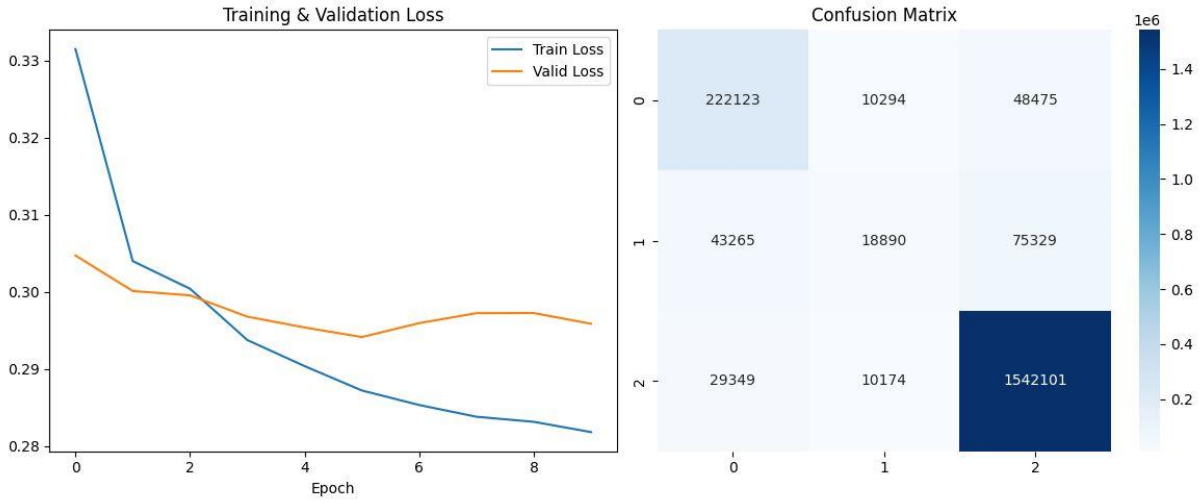

Figure 2: Confusion matrix and training history for the LSTM model

Discussion: The LSTM model demonstrated the highest performance across all key metrics, indicating its strength in capturing the complexities of sentiment in text. The CNN model, developed by Ramana, followed closely and offered an excellent balance between performance and training efficiency. The Naive Bayes model provided a solid baseline but lacked the nuance captured by deep learning architectures. The visualized training and validation losses for the LSTM indicate stable training without overfitting.

4. Summary and Conclusions

This project demonstrated the practical application of machine learning and NLP techniques to a large real-world dataset. The baseline model provided a useful benchmark, while the LSTM and CNN and RNN models added advanced performance suited for deep learning tasks. Through this project, I gained practical experience in pipeline construction, model design, optimization, and evaluation. Future iterations could benefit from fine-tuning transformer models or applying attention-based architectures.

5. Code Attribution

Generative AI tools were used for most of the code synthesis. No lines of code were copied directly from any external sources.

6. References

Intentionally omitted; no external references or direct code reuse involved.