Sentiment Analysis of Book Reviews Using RNN, LSTM, and Word2Vec Embeddings

Siman Giri

May 2025

Abstract

This project focuses on sentiment analysis of book reviews to predict ratings (1 to 5) using deep learning models, specifically Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM) networks, and LSTM with pre-trained Word2Vec embeddings. The dataset comprises 12,000 book reviews, preprocessed through text cleaning, tokenization, lemmatization, and stopword removal. Three models were developed: a Simple RNN, an LSTM, and an LSTM with Word2Vec embeddings. The models were trained and evaluated on accuracy, precision, recall, and F1-score, with the LSTM+Word2Vec model achieving the highest test accuracy of 47.83%, significantly outperforming the Simple RNN (25.00%) and LSTM (24.83%) models. The use of pre-trained Word2Vec embeddings improved performance by providing richer word representations. However, overall accuracies suggest challenges such as class imbalance and model complexity. Future work includes hyperparameter tuning, addressing class imbalance, and exploring advanced architectures like Transformers. This project demonstrates the potential of deep learning for sentiment analysis while highlighting areas for improvement.

1 Introduction

Sentiment analysis, a subfield of natural language processing (NLP), involves classifying text based on its emotional tone, such as positive, negative, or neutral. This project addresses the multiclass classification of book reviews to predict ratings from 1 (negative) to 5 (positive), a task with applications in e-commerce, recommendation systems, and customer feedback analysis. Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks are well-suited for sequential data like text, as they capture temporal dependencies. Additionally, pre-trained Word2Vec embeddings enhance word representations by leveraging semantic relationships learned from large corpora. Previous work in sentiment analysis has utilized RNNs and LSTMs for tasks like tweet classification [?], but combining these with Word2Vec embeddings remains underexplored for book reviews. This report details the dataset, methodology, experiments, and results of the project, comparing the performance of RNN, LSTM, and LSTM+Word2Vec models.

2 Dataset

The dataset, sourced from a CSV file named Book_review.csv, contains 12,000 book reviews. Each entry includes:

- Unnamed: 0: An index column (integer).
- rating: Integer rating from 1 to 5.
- reviewText: The full review text (string).
- summary: A brief summary of the review (string, with 2 missing values).

The dataset was preprocessed by filling missing summary values with empty strings and combining summary and reviewText into a single text column. Text preprocessing steps included:

- Expanding contractions (e.g., "can't" to "cannot").
- Converting text to lowercase.
- Removing URLs, emojis, punctuation, numbers, user mentions, hashtags, and duplicate words.
- Tokenizing text and removing stopwords.
- Applying lemmatization to reduce words to their base form.

The preprocessed text was stored in a cleaned_text column, preparing it for model input.

3 Methodology

3.1 Text Preprocessing

The text preprocessing pipeline ensured clean and standardized input data. Key steps included tokenization using NLTK's word_tokenize, stopword removal using NLTK's English stopword list, and lemmatization with WordNetLemmatizer. A custom function, text_cleaningHumanize, integrated these steps, producing tokenized and lemmatized text suitable for modeling.

3.2 Model Architecture

Three deep learning models were developed using TensorFlow's Keras API:

- Simple RNN: Consists of an Embedding layer (128 dimensions), a SimpleRNN layer (128 units, tanh activation), two Dropout layers (0.3), a Dense layer (64 units, ReLU), and a Dense output layer (5 units, softmax).
- LSTM: Includes an Embedding layer (128 dimensions), an LSTM layer (64 units), a Dense layer (32 units, ReLU), and a Dense output layer (5 units, softmax).
- LSTM+Word2Vec: Uses a non-trainable Embedding layer initialized with 300-dimensional Word2Vec embeddings (Google News), a Bidirectional LSTM layer (250 units), Dropout (0.2), a Flatten layer, and multiple Dense layers (128, 64, 32, 16 units, ReLU with Dropout), ending with a Dense output layer (5 units, softmax).

3.3 Loss Function, Optimizer, and Hyperparameters

All models used categorical_crossentropy as the loss function, suitable for multiclass classification. The adam optimizer was chosen for its adaptive learning rate. Key hyperparameters included:

- Batch size: 32.
- Epochs: Up to 15, with early stopping (patience=5, monitoring val_loss).
- Embedding dimensions: 128 (RNN, LSTM), 300 (LSTM+Word2Vec).
- Sequence length: 180 (95th percentile of training sequence lengths).

4 Experiments and Results

4.1 RNN vs. LSTM Performance

The Simple RNN and LSTM models were trained on padded sequences of length 180, with a vocabulary size of 31,750. The Simple RNN achieved a test accuracy of 25.00%, while the LSTM scored 24.83%. Both models exhibited poor performance, with confusion matrices showing a bias toward predicting higher ratings (e.g., class 4 for RNN, class 3 for LSTM). The LSTM+Word2Vec model significantly outperformed both, with a test accuracy of 47.83%, demonstrating better generalization across all classes.

4.2 Computational Efficiency

Training was conducted on Google Colab with GPU support. The Simple RNN trained faster (e.g., 33–46 seconds per epoch) due to its simpler architecture, while the LSTM required slightly more time (e.g., 4 seconds for inference on test data). The LSTM+Word2Vec model was the most computationally intensive, with a larger parameter count (9.5 million, mostly non-trainable) and longer inference time (4 seconds for test data). Memory usage was higher for LSTM+Word2Vec due to the 300-dimensional embeddings and bidirectional LSTM.

4.3 Training with Different Embeddings

The Simple RNN and LSTM used randomly initialized embeddings trained during model fitting, while the LSTM+Word2Vec model leveraged pre-trained Word2Vec embeddings from the Google News dataset. The Word2Vec embeddings improved performance by providing semantic context, resulting in a 47.83% accuracy compared to 25.00% (RNN) and 24.83% (LSTM). The pre-trained embeddings reduced the need for extensive training data, enhancing model robustness.

4.4 Model Evaluation

Evaluation metrics included:

- Accuracy: Overall classification accuracy on the test set (1200 samples).

- Confusion Matrix: Normalized matrices showed class-specific performance, with LSTM+Word2Vec balancing predictions better across classes.
- Precision, Recall, F1-Score: For LSTM+Word2Vec, macro-averaged precision was 0.46, recall 0.45, and F1-score 0.45, compared to 0.05/0.20/0.08 (RNN) and 0.09/0.20/0.08 (LSTM).

The classification report for LSTM+Word2Vec showed balanced performance, with F1-scores ranging from 0.23 (class 2) to 0.66 (class 4), indicating better handling of class variability.

5 Conclusion and Future Work

This project successfully implemented sentiment analysis of book reviews using RNN, LSTM, and LSTM+Word2Vec models. The LSTM+Word2Vec model outperformed others with a test accuracy of 47.83%, highlighting the advantage of pre-trained embeddings in capturing semantic relationships. However, the low accuracies of the Simple RNN (25.00%) and LSTM (24.83%) suggest limitations, including potential class imbalance, insufficient model capacity, and suboptimal hyperparameters. The models also showed biases toward higher ratings, as seen in confusion matrices.

Future work includes:

- Hyperparameter Tuning: Optimize learning rate, LSTM units, and dropout rates.
- Class Imbalance: Apply techniques like class weights or oversampling to balance rating distribution.
- Advanced Architectures: Explore Transformers or BERT for improved performance.
- Larger Datasets: Incorporate more diverse review data to enhance generalization.

The project underscores the potential of deep learning for sentiment analysis while identifying key areas for improvement, paving the way for more robust NLP applications.