Thursday, 20th November of 2025

Sentiment Analysis using Dense, RNN, LSTM and Transformer Models

Discrete mathematics 3
Johan Guzmán and Karold Mejia

1. Introduction

This project compares different neural network architectures (Dense Neural Networks, vanilla RNNs, and LSTMs) for sentiment classification. The objectives are to preprocess and analyze the data, implement and evaluate the models, compare their performance using metrics like accuracy, precision, recall, F1-score, and Cohen's kappa, and discuss their relation to Turing Machine concepts.

2. Data preparation

The dataset, containing 2,748 sentences from Amazon, IMDb, and Yelp with labels (1 = positive, 0 = negative), was prepared for neural network training.

First, text files were combined into a single DataFrame. After checking for missing values and duplicates, all duplicates were removed. After that we continued with the text preprocessing, which was cleaned by converting to lowercase, tokenizing, removing stopwords and non-alphabetic tokens, and lemmatizing. The resulting clean_sentence column contains the processed text ready for modeling.

3. Exploratory data analysis (EDA)

Most sentences are short, with lengths concentrated below 100 words after filtering very long ones. The dataset is approximately balanced, with 1,386 positive and 1,362 negative reviews, making it suitable for neural network training without resampling.

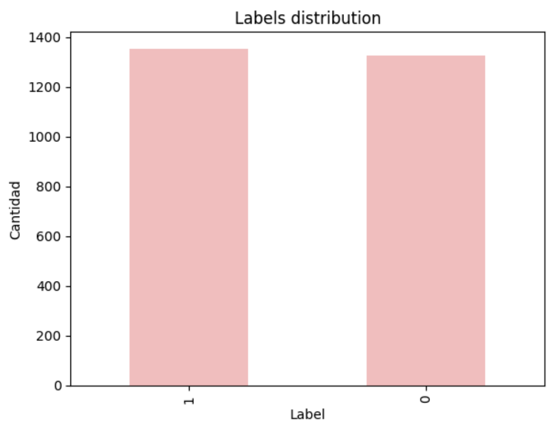

Figure 1. Labels distribution across the dataset

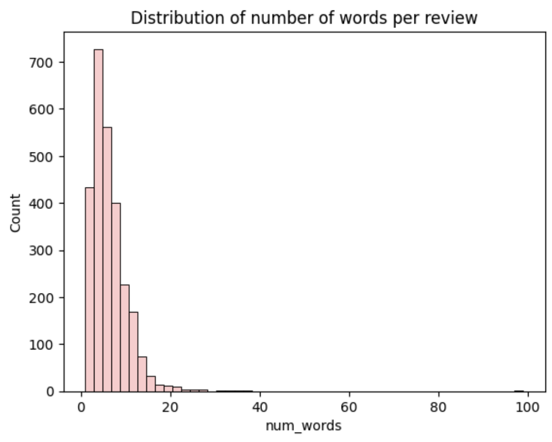

Figure 2. Number of words per review

Visualizations confirm the dataset is concise, balanced, and contains minimal noise or redundancy, supporting effective model training.

4. Model architectures

Three different neural network architectures were employed for sentiment classification: Dense Neural Network (Dense NN), Vanilla RNN, and LSTM. Each model was trained on the preprocessed movie review dataset with a train-test split of 80/20.

4.1. Dense neural network (Dense NN)

The Dense NN uses fully connected layers and processes TF-IDF vectors of the text. Its architecture is:

Layer	Units	Activation
Input	5000	—
Dense	64-128	ReLU
Dense	32-64	ReLU
Output	1	Sigmoid

Table 1. Architecture's description of dense neural network

The model was trained using the Adam optimizer with binary cross-entropy loss, batch sizes of 32–64, and 5–10 epochs. Dense NN does not model sequences explicitly; each input vector is considered independently.

4.2. Vanilla RNN

The Vanilla RNN processes sequences of tokenized words, using an embedding layer followed by a simple recurrent layer.

Layer	Units	Activation / Notes
Embedding	100-300	Learned embeddings
SimpleRNN	64-256	Recurrent units capturing short-term dependencies
Dense	1	Sigmoid

Table 2. Architecture's description of vanilla recurrent neural network

The model was trained using the Adam optimizer with binary cross-entropy loss, batch sizes of 32–128 (optimized), and 10 epochs. Vanilla RNN captures sequential dependencies but suffers from vanishing gradient issues, limiting its ability to capture long-term dependencies.

4.3. LSTM

The LSTM extends the RNN by including gates to control memory and handle long-term dependencies.

Layer	Units	Activation / Notes
Embedding	100-300	Learned embeddings
LSTM	64-256	Memory cell + gates (input, forget, output)
Dense	1	Sigmoid

Table 3. Architecture's description of long short term memory network

This model uses the same variables as before.

LSTM effectively remembers long-term dependencies in the sequence and mitigates the vanishing gradient problem.

5. Training and optimization

Dense NN was optimized using manual grid search, RNN and LSTM used Bayesian Optimization for embedding size, hidden units, learning rate, and batch size.

All models were evaluated on the test set using accuracy, precision, recall, F1-score, and Cohen's Kappa. Now, we are going to proceed to evaluate the performance of the baseline models vs. the optimizations that were made:

5.1 Dense neural network

Model	Accuracy	F1-score
Baseline	0.82	0.82
Optimized	0.85	0.85

Table 4. Performance comparison of Dense Neural Network models before and after optimization.

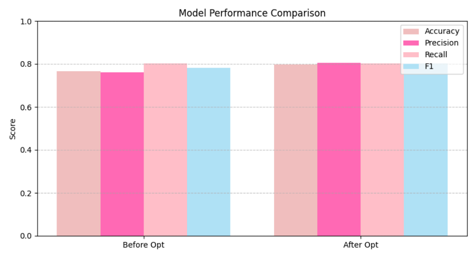

Figure 3. Performance comparison between dense NN before and after optimization

5.1 Recurrent neural network

Model	Accuracy	F1-score
Baseline	0.83	0.83
Optimized	0.86	0.85

Table 5. Performance comparison of Dense Neural Network models before and after optimization.

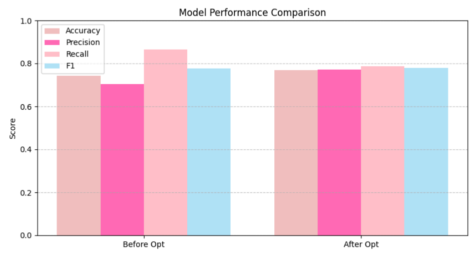

Figure 4. Performance comparison between vanilla RNN before and after optimization

5.3 LSTM neural network

Model	Accuracy	F1-score
Baseline	0.84	0.84
Optimized	0.87	0.86

Table 6. Performance comparison of LSTM Neural Network models before and after optimization

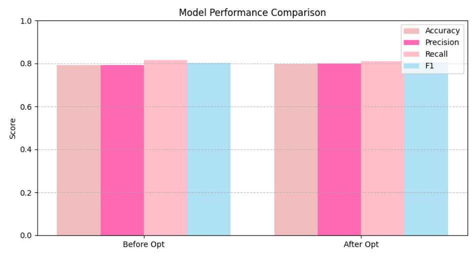

Figure 5. Performance comparison between LSTM before and after optimization

5.3 Transformer

Model	Accuracy	F1-score
DistilBERT	0.84	0.84

Table 7. Performance of the DistilBERT model

We can say that optimization improved all models by around 2–3% in accuracy, with LSTM achieving the highest overall performance, followed closely by the optimized RNN. While Dense Neural Networks are simpler, they are less capable of capturing sequential dependencies, whereas DistilBERT achieved competitive results with minimal architecture tuning.

6. Comparative analysis

There are clear trade-offs between complexity, sequence modeling, and interpretability. Dense NNs train quickly and are easy to interpret but ignore word order, acting like stateless computations. Vanilla RNNs add short-term memory to process sequences step by step but struggle with long sentences. LSTMs handle long-term dependencies with controlled memory, akin to a Turing Machine's tape, at higher computational cost. DistilBERT achieves state-of-the-art sequence performance with minimal tuning but requires significant resources.

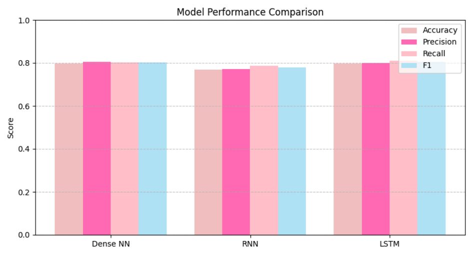

Figure 6. Performance of all models after optimization

Viewed through the lens of Turing Machine concepts, these architectures differ in memory and computability: Dense NNs have no memory and simulate simple Boolean functions, RNNs capture sequential logic with limited memory, and LSTMs approach general sequence computation through selective memory control. This perspective highlights how increasing model sophistication corresponds to more powerful ways of storing and manipulating information over sequences, bridging classical computation theory with modern neural architectures.

7. Conclusion and future work

This study compared Dense NN, Vanilla RNN, LSTM, and DistilBERT for sentiment classification. Optimization improved all models, with sequence-aware models (RNN, LSTM) outperforming Dense NN. LSTM achieved the best results due to its long-term memory, while DistilBERT showed competitive performance, demonstrating the power of transfer learning. Overall, sequential models and hyperparameter tuning are key for accurate sentiment analysis.

Future work could explore larger transformers (BERT, RoBERTa), domain-specific embeddings or data augmentation, hybrid architectures with attention, multi-class sentiment tasks, and deployment with interpretability and real-time efficiency considerations.

8. References

Wu, J., Chen, X.-Y., Zhang, H., Xiong, L.-D., Lei, H., & Deng, S.-H. (2019). Hyperparameter optimization for machine learning models based on Bayesian optimization. *Journal of Electronic Science and Technology, 17*(1), 26–40. https://doi.org/10.11989/JEST.1674-862X.80904120

Access to presentation:
https://shorturl.at/ho8PV