Tailored Preprocessing for Sentiment Analysis: A Comparative Study on IMDB 50K Reviews

Marko Milenovic
marko.milenovic@student.um.si
Faculty of Electrical Engineering and Computer Science,
University of Maribor
SI-2000 Maribor, Slovenia

Viktor Rackov
viktor.rackov@student.um.si
Faculty of Electrical Engineering and Computer Science,
University of Maribor
SI-2000 Maribor, Slovenia

Nemanja Mitric
nemanja.mitric@student.um.si
Faculty of Electrical Engineering and Computer Science,
University of Maribor
SI-2000 Maribor, Slovenia

Mario Bojarovski
mario.bojarovski@student.um.si
Faculty of Electrical Engineering and Computer Science,
University of Maribor
SI-2000 Maribor, Slovenia

ABSTRACT

This paper presents a comparative evaluation of sentiment classification models on the IMDB 50K dataset, spanning classical machine learning, deep learning, and transformer-based architectures. A key contribution is the implementation of model-specific preprocessing pipelines, designed to align with the structural and learning characteristics of each model family. Classical models were paired with extensive normalization, while deep learning and transformer models used minimal preprocessing focused on noise reduction.

The results demonstrate that tailoring preprocessing to model architecture leads to consistent improvements in performance across all categories. In addition to quantitative evaluation using accuracy, macro F1 score, and ROC AUC, we analyzed training behavior, confusion patterns, and model interpretability. These findings underscore the importance of preprocessing as a critical design choice in sentiment classification pipelines.

KEYWORDS

Sentiment Analysis, IMDB 50K, Preprocessing, Machine Learning, Deep Learning, Transformers, XGBoost, Naive Bayes, RNN, CNN, LSTM, GRU

1 INTRODUCTION

Natural Language Processing (NLP) enables computers to interpret and generate human language, powering applications from machine translation to question answering [6]. A core NLP task is sentiment analysis, which seeks to identify and classify opinions or emotions expressed in text as positive, negative, or neutral. This capability is invaluable for monitoring customer feedback, analyzing social media, and guiding content recommendations.

High-quality annotated data is essential for training and evaluating sentiment classifiers. The IMDB 50K movie reviews dataset —50,000 evenly split positive and negative user reviews—has become a standard benchmark due to its size, balance, and public availability [4]. Researchers have used this corpus to test a broad spectrum of methods, from simple lexicon lookups to cutting-edge neural networks.

Sentiment analysis techniques can be grouped into four families:

- *Lexicon-based methods* rely on predefined dictionaries of positive and negative words, scoring texts by summing individual term polarities [11].
- *Traditional machine learning* converts documents into feature vectors (e.g. bag-of-words, TF–IDF) [10] and trains classifiers such as Logistic Regression [12], Support Vector Machines (SVM) [3], Multinomial Naive Bayes [1], and XGBoost [2].
- *Deep learning architectures* learn representations end-to-end from raw or embedded text. Convolutional Neural Networks (CNNs) capture local n-gram patterns via sliding filters [9], Recurrent Neural Networks (RNNs) process sequences token by token, Long Short-Term Memory (LSTM) [8] and Gated Recurrent Unit (GRU) [14] networks add gating to remember long-range dependencies, and their bidirectional variants (BiLSTM, BiGRU) incorporate both past and future context.
- *Transformer-based models* such as BERT [13] and DistilBERT [7] employ self-attention to model all token interactions simultaneously and are pretrained on massive text corpora before fine-tuning on sentiment tasks.

We conduct a comprehensive evaluation of twelve diverse sentiment classification models on the IMDB 50K dataset, spanning classical machine learning, deep learning, and transformer-based approaches.

- **Logistic Regression**: a linear classifier estimating class probabilities via a sigmoid over feature weights.
- **SVM**: finds the maximum-margin hyperplane separating positive and negative examples in high-dimensional feature space.
- **Multinomial Naive Bayes**: a probabilistic model assuming feature independence, estimating per-class word frequencies.
- **XGBoost**: gradient-boosted decision trees that iteratively correct previous residuals for high predictive power.
- **CNN**: applies convolutional filters to word embeddings to detect informative phrases, followed by pooling.
- **RNN**: maintains a hidden state through the sequence to capture context dynamically.

- **LSTM / GRU**: enhance RNNs with gated cells to preserve long-term information and mitigate vanishing gradients.
- **BiLSTM / BiGRU**: process text in both forward and backward directions to enrich token representations.
- **BERT / DistilBERT**: pretrained transformer encoders fine-tuned for sentiment classification, with DistilBERT offering a lighter, faster variant.

To ensure a fair and comprehensive comparison across all model families, we evaluate model performance using three well established classification metrics: accuracy, macro-averaged F1 score, and the area under the Receiver Operating Characteristic curve (ROC AUC). Additionally, we assess computational efficiency by plotting each model's predictive performance against its training duration, allowing us to visualize the trade-off between effectiveness and computational cost.

We define the key terms used throughout this section as follows: true positives (TP) are correctly predicted positive samples, true negatives (TN) are correctly predicted negative samples, false positives (FP) are negative samples incorrectly predicted as positive, and false negatives (FN) are positive samples incorrectly predicted as negative.

The metrics are computed as follows:

- **Accuracy** – the overall proportion of correct predictions:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Macro-F1 Score** – the unweighted average of F1 scores for each class, where F1 is the harmonic mean of precision and recall:

$$\text{Precision} = \frac{TP}{TP + FP}$$
$$\text{Recall} = \frac{TP}{TP + FN}$$
$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$
$$\text{Macro-F1} = \frac{\text{F1}_{\text{Positive}} + \text{F1}_{\text{Negative}}}{2}$$

  This metric is especially relevant when both false positives and false negatives are equally undesirable, ensuring balanced performance across classes.
- **ROC AUC** – the area under the Receiver Operating Characteristic curve, which plots the true positive rate (TPR) against the false positive rate (FPR) across all classification thresholds:

$$\text{TPR (Recall)} = \frac{TP}{TP + FN}$$
$$\text{FPR} = \frac{FP}{FP + TN}$$

  A higher ROC AUC indicates stronger model discrimination capability—its ability to rank positive instances higher than negative ones, regardless of the threshold.
- **Training Efficiency (Performance vs. Time)** – Rather than reporting raw training durations, we assess computational efficiency by plotting ROC AUC against training time. This visualization provides an intuitive representation of the trade-offs between performance and resource usage, enabling clearer insight into which models deliver the best return on computational investment.

In addition to quantitative evaluation, we perform a qualitative analysis of model misclassifications to gain deeper insight into common failure cases and limitations that are not captured by metrics alone. These analyses help identify patterns in prediction errors and guide future improvements in model design and preprocessing.

The goal of this research is to replicate and improve upon the results of Amulya et al. [5] on sentiment classification of IMDB movie reviews. While their study utilized classical machine learning and a limited set of deep learning models with uniform preprocessing, our work aims to achieve superior performance—measured by accuracy, macro-F1 score, and ROC AUC—through two key enhancements.

First, we implement a preprocessing pipeline tailored to each model family: extensive normalization for classical models, minimal cleaning for deep learning architectures, and tokenizer-aligned input formatting for transformer-based models. Second, we extend the model set by incorporating more expressive variants, including bidirectional recurrent networks (BiLSTM and BiGRU) and transformer-based architectures (BERT and DistilBERT). These improvements ensure that model inputs are better aligned with each architecture's strengths, resulting in more accurate and robust sentiment classification compared to the original benchmarks.

2 METHODOLOGY

This section outlines the methodology used for sentiment analysis on the IMDB 50K dataset. The implementation is structured in two phases. In the first phase, we train classical machine learning models using feature-engineered representations such as Bag-of-Words and TF–IDF, alongside standard deep learning architectures including convolutional neural networks (CNN), simple recurrent networks (RNN), and gated recurrent units (GRU, LSTM). In the second phase, we expand this baseline by implementing bidirectional variants of the recurrent models (BiLSTM and BiGRU) and fine-tuning transformer-based architectures such as BERT and DistilBERT, which offer deeper contextual representation and improved performance.

2.1 Preprocessing Pipeline

In previous research by [5], preprocessing typically involved general text normalization steps such as lowercasing, removal of noisy or special characters, stemming, and stopword filtering. Features were then extracted using CountVectorizer or TF–IDF, which form the foundation for traditional text classification pipelines.

In our work, we adopt a more *tailored preprocessing strategy* that takes into account the informal and often slanged nature of IMDB movie reviews, and is designed specifically to align with the characteristics of each model family.

For classical machine learning models (Logistic Regression, SVM, Naive Bayes, XGBoost), we apply a preprocessing pipeline that captures sentiment-relevant patterns and mitigates noise in user-generated content. This includes:

- Contraction expansion (e.g., `don't` → `do not`) to normalize informal expressions,
- Emoticon mapping (e.g., `:)` → `smile`) to preserve emotional indicators often used in reviews,

- Negation tagging (e.g., `not good` → `not_good`) to retain semantic inversion patterns,
- Removal of HTML tags, URLs, and standalone numbers, with punctuation normalized to reduce inconsistencies,
- Lemmatization using WordNet to unify word forms,
- Stopword filtering with NLTK, while keeping negation terms (`no, not, never, hardly`) due to their importance in sentiment classification.

These steps are especially effective for models relying on sparse, frequency-based features, which benefit from cleaner and semantically enriched inputs.

For deep learning models (CNN, RNN, LSTM, GRU, BiGRU, BiLSTM), we use minimal preprocessing, limited to the removal of HTML tags and URLs. This decision reflects the ability of neural networks to learn contextual and structural patterns directly from raw token sequences. Aggressive cleaning such as stemming or stopword removal can obscure valuable signals, so the raw form of the input is largely preserved and passed to a `TextVectorization` layer with a vocabulary size of 10,000 and a sequence length of 400 tokens.

For transformer-based models (BERT and DistilBERT), we likewise apply only minimal preprocessing—removing HTML and URLs—before passing the text to pretrained tokenizers (`bert-base-uncased` or `distilbert-base-uncased`). These tokenizers handle casing, subword splitting, and sequence padding in a way that is consistent with the model's pretraining. Introducing custom normalization at this stage could disrupt that alignment, so we intentionally avoid additional modifications and allow the model to fully leverage the contextual richness of original inputs.

This model-specific preprocessing approach ensures that each architecture receives input that aligns with its learning mechanism and optimally handles the informal style and linguistic variability present in IMDB reviews.

2.2 Classical Machine Learning Models

Four classical machine learning models were implemented: Logistic Regression, Multinomial Naive Bayes, Support Vector Machine (SVM), and XGBoost.

Logistic Regression was configured with L2 regularization to prevent overfitting and was trained using the `liblinear` solver. To handle class imbalance, class weights were set to be balanced automatically.

Multinomial Naive Bayes was used with default parameters, as it is well-suited for text classification tasks involving discrete feature inputs such as word counts or term frequencies.

Support Vector Machine was implemented using the LinearSVC variant with a regularization parameter $C = 1.0$. This linear classifier is efficient for high-dimensional spaces and was chosen for its scalability with sparse data.

XGBoost, a gradient-boosted decision tree ensemble, was also included. It was used with default settings to maintain a balance between model complexity and training efficiency.

2.3 Deep Learning Architectures

We implemented six deep learning models for binary sentiment classification: Convolutional Neural Network (CNN), Simple Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), Bidirectional LSTM (BiLSTM), Gated Recurrent Unit (GRU), and Bidirectional GRU (BiGRU). All models used integer-encoded sequences derived from a `TextVectorization` layer with a vocabulary size of 10,000 and a fixed sequence length of 400 tokens. Each token was embedded into a 128-dimensional dense vector via an embedding layer shared across models.

The CNN model consisted of a 1D convolutional layer with 128 filters and a kernel size of 5, followed by a global max pooling layer to extract the most salient features across the sequence. A dense hidden layer with ReLU activation and a dropout layer with a rate of 0.5 were added before the final sigmoid output layer.

The RNN model utilized a `SimpleRNN` layer with 64 hidden units, combined with dropout regularization. The LSTM model followed the same structure but used an `LSTM` layer, enabling better memory retention for long-term dependencies. The GRU model was structurally similar, replacing the recurrent unit with a GRU cell, which has fewer parameters than LSTM but retains gating mechanisms for effective sequence modeling.

Bidirectional variants of the LSTM and GRU models were also constructed. These wrapped the recurrent layers in a bidirectional wrapper, allowing the models to process sequences in both forward and backward directions. This design helps capture dependencies and contextual clues that may be missed in a unidirectional setup.

All models included a dense hidden layer with 64 units and ReLU activation after the recurrent or convolutional component, followed by dropout for regularization and a sigmoid output neuron for binary classification. Training was performed for a maximum of 10 epochs using the Adam optimizer and binary crossentropy loss. Early stopping was applied with a patience of 3 epochs based on validation loss, and the best model weights were restored automatically to prevent overfitting.

2.4 Transformer-Based Models

We fine-tuned two transformer-based models for sentiment classification: BERT and DistilBERT. These models were selected for their strong ability to capture contextual relationships in text and their proven success in natural language understanding tasks.

The training was carried out using the Hugging Face Transformers library and the built-in Trainer API. Both models were fine-tuned over 2 epochs with a learning rate of 2e-5, weight decay of 0.01, and a batch size of 16. The training process included automated evaluation on a validation split.

This setup enabled the models to adapt their general-purpose language representations to the specific task of sentiment classification, achieving strong results with minimal changes to their pretrained structure.

2.5 Evaluation Metrics

To assess model performance, we report three well-established evaluation metrics: Accuracy, Macro F1 Score, and ROC AUC (Receiver Operating Characteristic Area Under Curve).

Accuracy measures the overall proportion of correctly classified reviews. Macro F1 Score computes the harmonic mean of precision and recall for each class independently, then averages the results, ensuring that both positive and negative classes are treated equally. ROC AUC quantifies the model's ability to distinguish between positive and negative sentiment across various classification thresholds, providing a threshold-independent evaluation of ranking quality.

Rather than listing raw training durations, we analyze computational efficiency by plotting model performance against training time. This allows us to highlight trade-offs between predictive power and resource consumption, particularly relevant for real-world applications. All models were trained on Google Colab's T4 GPU.

To complement quantitative metrics, we also perform interpretability and error analyses. For classical models, we inspect learned coefficients (e.g., word weights in Logistic Regression) to identify which features are most influential in sentiment prediction. Additionally, we generate confusion matrices for the best and the worst model to visualize prediction errors and identify common misclassification patterns. Finally, we conduct a qualitative analysis of misclassified examples to uncover potential limitations, such as difficulty handling negation, ambiguous phrasing, or mixed sentiment, which may not be captured by numerical metrics alone.

3 RESULTS

In this section, we present and analyze the performance of classical machine learning models (Logistic Regression, Multinomial Naive Bayes, SVM, XGBoost), deep learning architectures (CNN, RNN, LSTM, GRU, BiGRU, BiLSTM), and transformer-based models (BERT, DistilBERT) on the IMDB 50K sentiment classification task.

Table 1 reports the performance of all models based on accuracy, macro-averaged F1 score, and ROC AUC. Among transformer-based models, BERT consistently outperformed all others, achieving the highest accuracy (93.35%), macro-F1 score (93.35%), and ROC AUC (0.9800). This strong performance is supported by both the model architecture and the minimal, tokenizer-consistent preprocessing that allowed BERT to make full use of contextual signals.

Within the deep learning category, GRU achieved the highest performance, slightly surpassing its bidirectional counterpart. CNN also performed competitively, benefiting from its ability to detect informative local patterns. Among classical models, Linear SVC with TF–IDF features reached 91.64% accuracy and a ROC AUC of 0.9740, showcasing that classical methods, when combined with sentiment-aware preprocessing (e.g., contraction expansion, negation tagging), can yield highly effective and computationally efficient models.

To offer a point of comparison, Table 2 reproduces results from the original study by Amulya et al. [5], who reported precision, recall, and F1 score metrics for their models. Although their evaluation setup differs slightly, particularly in the absence of ROC AUC, it is clear that our tailored preprocessing and expanded model suite lead to significantly improved classification performance.

Note: The above results from Amulya et al. [5] use slightly different evaluation metrics. For comparability, only accuracy and F1 are referenced, though our study additionally reports macro F1 and ROC AUC. Our superior results are largely attributed to the model-specific preprocessing strategies and architectural extensions introduced in this work.

Table 1: Performance metrics for all models (our implementation)

Model	Accuracy	Macro F1	ROC AUC
Classical Machine Learning Models			
MultinomialNB (TF–IDF)	0.8945	0.8945	0.9600
MultinomialNB (Bag-of-Words)	0.8891	0.8891	0.9470
Logistic Regression (TF–IDF)	0.9109	0.9109	0.9680
Logistic Regression (Bag-of-Words)	0.9077	0.9077	0.9650
Linear SVC (TF–IDF)	**0.9164**	**0.9164**	**0.9740**
Linear SVC (Bag-of-Words)	0.9014	0.9014	0.9620
XGBoost (TF–IDF)	0.8671	0.8671	0.9420
XGBoost (Bag-of-Words)	0.8696	0.8696	0.9440
Deep Learning Models			
CNN	0.9008	0.9008	0.9650
RNN	0.7803	0.7802	0.8840
LSTM	0.8912	0.8895	0.9530
BiLSTM	0.8848	0.8852	0.9580
GRU	**0.9047**	**0.9031**	**0.9630**
BiGRU	0.9003	0.9000	0.9640
Transformer-Based Models			
DistilBERT	0.8929	0.8929	0.9590
BERT	**0.9335**	**0.9335**	**0.9800**

Table 2: Performance metrics from Amulya et al. [5] (reproduced)

Model	Precision	Recall	F1 Score	Accuracy
TF–IDF				
Logistic Regression	0.89	0.85	0.87	0.86
SVM	0.89	0.86	0.86	0.87
Naive Bayes	0.87	0.85	0.86	0.86
XGBoost	0.84	0.73	0.86	0.86
CountVectorizer				
Logistic Regression	0.87	0.86	0.87	0.86
SVM	0.86	0.86	0.86	0.86
Naive Bayes	0.87	0.85	0.86	0.86
XGBoost	0.84	0.74	0.79	0.81
Deep Learning				
CNN	0.94	0.85	0.87	0.87
RNN	0.95	0.86	0.88	0.88
LSTM	0.72	0.70	0.71	0.71

Performance vs. Training Time. While transformer-based models like BERT deliver the highest predictive performance, they also incur significantly longer training times compared to classical and deep learning models. Figure 1 visualizes this trade-off by plotting model accuracy against training duration. Classical models such as Logistic Regression and Linear SVC provide a compelling balance between efficiency and accuracy, while GRU and CNN models offer competitive results with moderate resource requirements. BERT, though the most accurate, is also the most computationally expensive, which is a crucial consideration in real-world deployment scenarios.

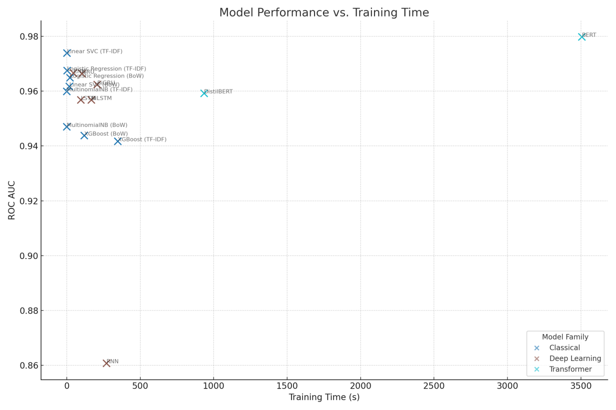

Figure 1: Model accuracy vs. training time

Loss Curve Analysis. Figure 2 shows training and validation loss curves for the deep learning models. Most architectures exhibit some degree of overfitting: validation loss initially declines but then rises in models like GRU, BiGRU, BiLSTM, and CNN. The RNN model, in particular, displays unstable and highly fluctuating validation loss. To mitigate overfitting, early stopping based on validation loss was applied, with a patience of three epochs and automatic restoration of the best model weights.

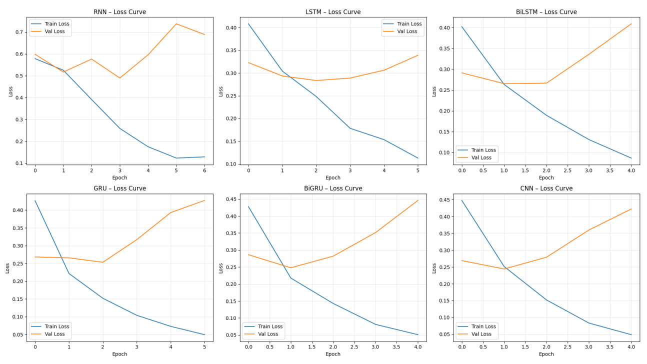

Figure 2: Training and validation loss curves for deep learning models

Confusion Matrix Comparison. Figure 3 presents the confusion matrices for BERT and the RNN model, representing the best- and worst-performing architectures, respectively. BERT misclassified 665 out of 10,000 samples (350 false positives and 315 false negatives), while the RNN misclassified 2,197 samples (991 false positives and 1,206 false negatives), highlighting a significant gap in reliability between these models.

ROC Curve Analysis. As illustrated in Figure 4, BERT achieves the highest ROC curve, reflecting its superior ability to distinguish between positive and negative sentiment. In contrast, the RNN shows the lowest AUC. The remaining models, including GRU, CNN, BiLSTM, and BiGRU, fall between these two extremes, consistent with the metrics in Table 1.

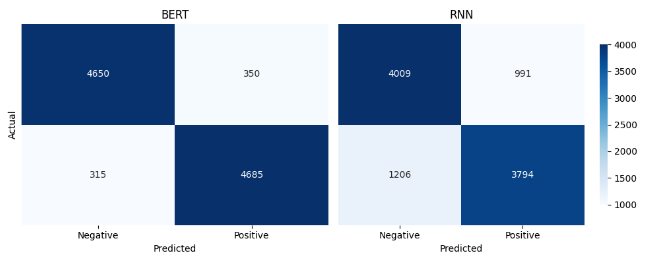

Figure 3: Confusion matrices for the best model (BERT, left) and the worst model (RNN, right) on the IMDB test set

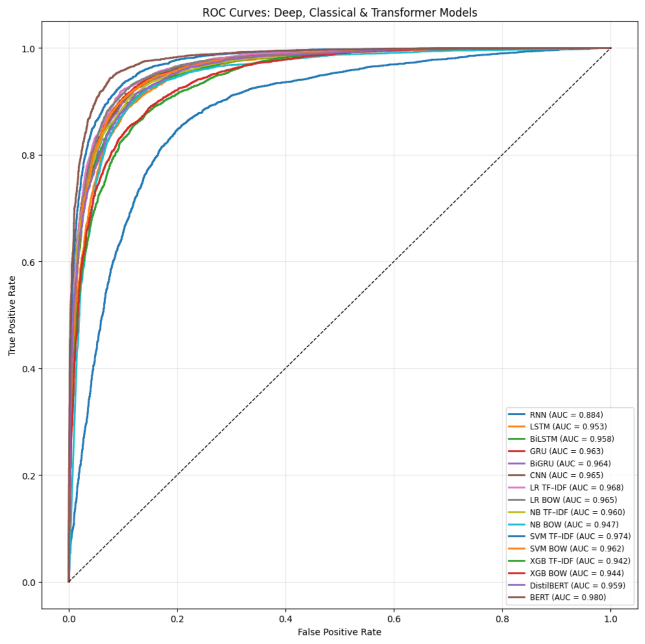

Figure 4: ROC curves comparing classification performance across classical, deep, and transformer models

Feature Importance. Figure 5 displays the top ten features with the highest absolute weights in the Logistic Regression model trained on TF–IDF representations. These words closely align with expected sentiment polarities—such as *excellent*, *wonderful*, *boring*, and *waste*—validating the model's ability to capture linguistically meaningful indicators of sentiment.

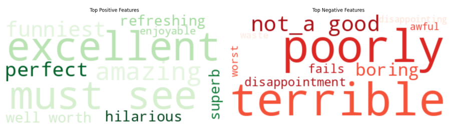

Figure 5: Top weighted words for the TF–IDF Logistic Regression model. Green bars indicate positive sentiment; red bars indicate negative sentiment.

Misclassification Analysis. An examination of BERT's misclassified IMDB reviews uncovers systematic weaknesses despite its strong overall performance. First, mixed-polarity sentences (e.g. "diverting, but not great") often lead BERT to latch onto the initial positive cue and ignore later criticism. Second, purely descriptive or behind-the-scenes passages with sparse sentiment (e.g. plot synopses, production anecdotes) are frequently assigned a default positive label. Third, negation and contrastive constructions ("wasn't great", "but then…") are sometimes mis-scoped, causing polarity flips. Fourth, sarcastic or ironic remarks—where praise is used to mock—confuse the model's literal interpretation of positive lexemes. Finally, very long or complex reviews can overwhelm the single-[CLS] classifier, which may lose track of the overall sentiment amid abundant side details. Addressing these failure modes—through targeted data augmentation, improved negation handling, and hierarchical encoding—should reduce BERT's residual error rate.

4 CONCLUSION

This study presented a comprehensive evaluation of sentiment classification models applied to the IMDB 50K dataset, encompassing classical machine learning, deep learning, and transformer-based approaches. A key contribution of this work lies in the design and application of model-specific preprocessing pipelines, which aligned data preparation strategies with the learning mechanisms of each model family.

Rather than relying on a uniform preprocessing scheme, we tailored the input pipeline to maximize each model type's effectiveness: classical models benefited from linguistic normalization and sentiment-aware features; deep learning architectures required minimal interference to preserve raw sequence patterns; and transformer models operated directly on tokenizer-aligned inputs. This alignment allowed even simple architectures to perform competitively, with classical models like Linear SVC achieving over 91

While transformer-based models such as BERT achieved the highest overall scores in accuracy and ROC AUC, our results demonstrate that performance gains were not solely a result of architecture. The preprocessing strategy consistently improved outcomes across all model categories and highlighted that thoughtful input design can rival the benefits of architectural complexity.

Beyond quantitative metrics, we incorporated interpretability analyses—examining loss curves, confusion matrices, and feature importance—to gain deeper insights into model behavior and misclassification patterns. These findings reinforce the importance of preprocessing as an essential design decision in sentiment analysis pipelines.

Future research could extend this approach to multilingual sentiment tasks, domain-specific applications, or zero-shot settings, where the interaction between preprocessing and model performance remains a critical but often underexplored dimension

REFERENCES

[1] Muhammad Abbas, K Ali Memon, A Aleem Jamali, Saleemullah Memon, and Anees Ahmed. 2019. Multinomial Naive Bayes classification model for sentiment analysis. *IJCSNS Int. J. Comput. Sci. Netw. Secur* 19, 3 (2019), 62.

[2] Khansa Afifah, Intan Nurma Yulita, and Indra Sarathan. 2021. Sentiment analysis on telemedicine app reviews using xgboost classifier. In *2021 international conference on artificial intelligence and big data analytics*. IEEE, 22–27.

[3] Munir Ahmad, Shabib Aftab, and Iftikhar Ali. 2017. Sentiment analysis of tweets using svm. *Int. J. Comput. Appl* 177, 5 (2017), 25–29.

[4] Nehal Mohamed Ali, Marwa Mostafa Abd El Hamid, and Aliaa Youssif. 2019. Sentiment analysis for movies reviews dataset using deep learning models. *International Journal of Data Mining & Knowledge Management Process (IJDKP) Vol* 9 (2019).

[5] K Amulya, SB Swathi, P Kamakshi, and Y Bhavani. 2022. Sentiment analysis on imdb movie reviews using machine learning and deep learning algorithms. In *2022 4th international conference on smart systems and inventive technology (ICSSIT)*. IEEE, 814–819.

[6] KR1442 Chowdhary and KR Chowdhary. 2020. Natural language processing. *Fundamentals of artificial intelligence* (2020), 603–649.

[7] Varun Dogra, Aman Singh, Sahil Verma, Kavita, NZ Jhanjhi, and MN Talib. 2021. Analyzing distilbert for sentiment classification of banking financial news. In *Intelligent Computing and Innovation on Data Science: Proceedings of ICTIDS 2021*. Springer, 501–510.

[8] Dan Li and Jiang Qian. 2016. Text sentiment analysis based on long short-term memory. In *2016 First IEEE International Conference on Computer Communication and the Internet (ICCCI)*. IEEE, 471–475.

[9] Shiyang Liao, Junbo Wang, Ruiyun Yu, Koichi Sato, and Zixue Cheng. 2017. CNN for situations understanding based on sentiment analysis of twitter data. *Procedia computer science* 111 (2017), 376–381.

[10] AMIT PURUSHOTTAM Pimpalkar and R Jeberson Retna Raj. 2020. Influence of pre-processing strategies on the performance of ML classifiers exploiting TF-IDF and BOW features. *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal* 9, 2 (2020), 49.

[11] Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Computational linguistics* 37, 2 (2011), 267–307.

[12] Mayur Wankhade, A Chandra Sekhara Rao, Suresh Dara, and Baijnath Kaushik. 2017. A sentiment analysis of food review using logistic regression. *Int J Sci Res Comput Sci Eng InformTechnol* 2, 7 (2017), 251–260.

[13] Hu Xu, Bing Liu, Lei Shu, and Philip S Yu. 2019. BERT post-training for review reading comprehension and aspect-based sentiment analysis. *arXiv preprint arXiv:1904.02232* (2019).

[14] Muhammad Zulqarnain, Rozaida Ghazali, Muhammad Aamir, and Yana Mazwin Mohmad Hassim. 2024. An efficient two-state GRU based on feature attention mechanism for sentiment analysis. *Multimedia Tools and Applications* 83, 1 (2024), 3085–3110.