Sentiment Analysis of IMDB Movie Reviews: A Comparative Study of Machine Learning and Deep Learning Approaches

Mete Harun Akcay

I. Introduction

In today's digital age, vast amounts of textual data are generated daily through social media, product reviews, and online forums. Sentiment analysis, a subfield of Natural Language Processing (NLP), is utilized to determine the emotional tone of a given text. It is widely applied in various domains, including customer feedback analysis, brand reputation monitoring, and recommendation systems. One practical application is in the film industry, where sentiment analysis assists potential viewers in deciding whether to watch a movie based on reviews, particularly for those who prefer quick insights rather than reading long and detailed reviews.

In this study, a sentiment classification model is developed to predict whether a given IMDB movie review is positive or negative. To achieve this, both traditional machine learning (ML) methods and deep learning (DL) approaches are examined. Specifically, the performance of the following models is compared:

- Naïve Bayes and Support Vector Machines (SVM) as ML-based classifiers.
- Long Short-Term Memory (LSTM) networks as a deep learning approach.

II. Data Analysis & Processing

The dataset used in this study is obtained from Hugging Face, containing a total of 50,000 IMDB movie reviews. The dataset is evenly distributed, with 25,000 reviews labeled as positive and 25,000 reviews labeled as negative, which is presented in Figure 1.

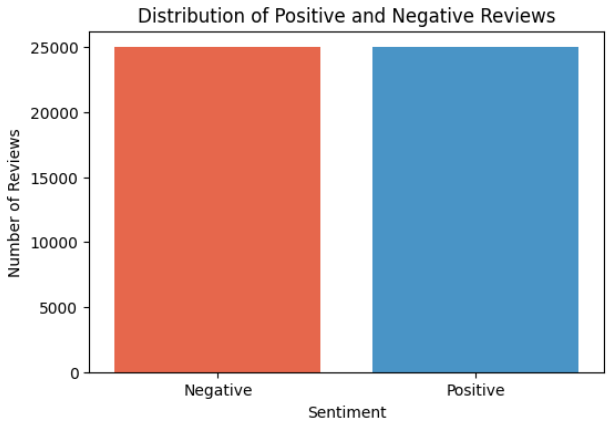

Fig. 1: Distribution of positive and negative reviews in the dataset

Before training the models, the text data was preprocessed to remove noise and standardize the format. The following steps were applied to each review:

1) **Removing HTML Tags:** Since some reviews contained HTML elements, the `BeautifulSoup` library was used to extract plain text.
2) **Lowercasing:** All text was converted to lowercase to ensure uniformity and reduce vocabulary size.
3) **Removing Punctuation and Numbers:** Non-alphabetic characters, including punctuation and numbers, were removed to focus only on meaningful words.
4) **Single Character Removal:** Isolated single characters (e.g., "a", "I") were removed as they contributed little to sentiment classification.
5) **Removing Extra Spaces:** Multiple consecutive spaces were replaced with a single space to maintain a clean text structure.
6) **Stopword Removal:** Common stopwords (e.g., "the", "is", "and") were filtered out using the NLTK stopwords list, as they did not contribute significantly to sentiment determination.
7) **Lemmatization:** Each word was reduced to its base form using WordNet lemmatization to handle different word variations (e.g., "running" $\rightarrow$ "run").

This preprocessing ensured that the text data was clean and consistent, improving model performance by reducing irrelevant variations in input.

An example of the preprocessing steps applied to a movie review is given below:

> **Original Review:** Basically there's a family where a little boy (Jake) thinks there's a zombie in his closet & his parents are fighting all the time.¡br /¿¡br /¿ This movie is slower than a soap opera... and suddenly, Jake decides to become Rambo and kill the zombie.¡br /¿¡br /¿ OK, first of all when you're going to make a film you must Decide if its a thriller or a drama! As a drama the movie is watchable. Parents are divorcing & arguing like in real life. And then we have Jake with his closet which totally ruins all the film! I expected to see a BOOGEYMAN similar

movie, and instead I watched a drama with some meaningless thriller spots.¡br /¿¡br /¿ 3 out of 10 just for the well playing parents & descent dialogs. As for the shots with Jake: just ignore them.

Cleaned Review: basically family little boy jake think zombie closet parent fighting time movie slower soap opera suddenly jake decides become rambo kill zombie ok first going make film must decide thriller drama drama movie watchable parent divorcing arguing like real life jake closet totally ruin film expected see boogeyman similar movie instead watched drama meaningless thriller spot well playing parent descent dialog shot jake ignore

III. Modeling

A. Initialization

After the preprocessing steps, the dataset was split into training and testing sets, with 85% of the data used for training and 15% for testing.

For the machine learning models (SVM and Naïve Bayes), text representations were generated using the Term Frequency-Inverse Document Frequency (TF-IDF) method. A TF-IDF vectorizer was initialized with a vocabulary size of 5000 words, capturing both unigrams and bigrams. The vectorizer was fitted on the training data and then applied to transform both the training and testing datasets into numerical representations.

For the LSTM model, a tokenizer was initialized with a vocabulary size of 10,000 words, and an out-of-vocabulary token was specified. The tokenizer was fitted on the training data, after which text sequences were converted into numerical representations. To ensure consistency in input length, sequences were padded and truncated to a maximum length of 200 words.

B. Training

For Naïve Bayes, hyperparameter tuning was performed using a grid search with cross-validation to determine the optimal smoothing parameter α. The best value was found to be $\alpha = 10$, which was used to train the final model.

Similarly, a linear SVM was trained, where the regularization parameter C was optimized through grid search. The best-performing value was found to be $C = 0.1$, and the model was trained using this parameter.

For the LSTM model, an embedding layer was followed by an LSTM layer with 128 units, incorporating dropout regularization to prevent overfitting. A dense output layer with a sigmoid activation function was used for binary classification. The model was compiled with binary cross-entropy loss and the Adam optimizer. Early stopping was applied to halt training if validation loss did not improve for three consecutive epochs. The model was trained for up to 15 epochs with a batch size of 64.

IV. Results

The final test accuracy for each model is presented in Table I.

Model	Final Test Accuracy
Naïve Bayes	0.8638
SVM	0.8912
LSTM	0.8754

TABLE I: Final test accuracy of Naïve Bayes, SVM, and LSTM models.

Among the three models, SVM achieved the highest accuracy on the test set, with a value of 0.8912. The LSTM model performed slightly worse, with an accuracy of 0.8754, while the Naïve Bayes classifier obtained the lowest accuracy at 0.8638.

A useful insight from the Naïve Bayes model is the relative importance of words in predicting sentiment. Figure 2 presents the top words that contributed most to positive and negative classifications.

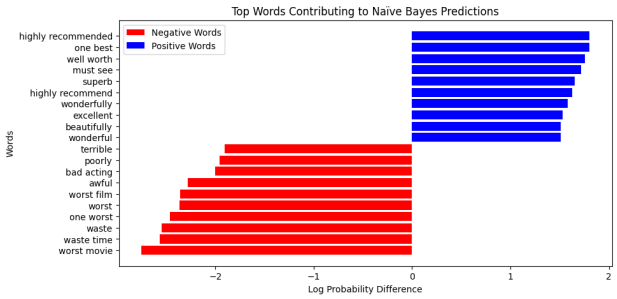

Fig. 2: Top words contributing to Naïve Bayes predictions.

As shown, words such as ”highly recommended,” ”wonderfully,” and ”excellent” strongly indicate positive sentiment, whereas words like ”worst movie,” ”waste time,” and ”awful” are associated with negative reviews, which makes total sense.

A similar graph showing the impact of individual words on the SVM model's predictions is visualized in Figure 3. As seen, words such as ”excellent,” ”great,” and ”amazing” strongly influence positive classifications, whereas words like ”worst,” ”awful,” and ”boring” push predictions toward the negative class.

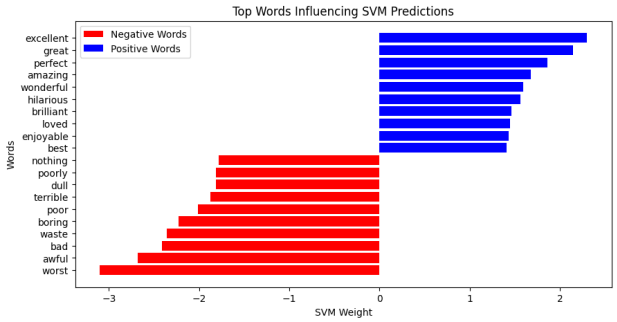

Fig. 3: Top words contributing to SVM predictions.

To further analyze the performance of the LSTM model, the training and validation accuracy and loss curves are shown in Figures 4 and 5, respectively.

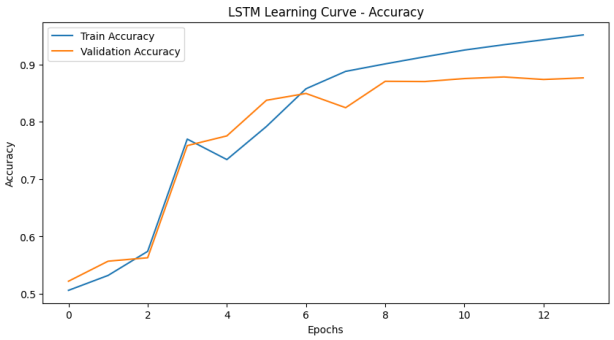

Fig. 4: Training and validation accuracy over epochs for the LSTM model.

Fig. 5: Training and validation loss over epochs for the LSTM model.

From Figure 4, it can be observed that during the first three epochs, the accuracy remained between 0.50 and 0.55. After epoch 3, a sudden increase occurred, and the accuracy reached approximately 0.70. The peak validation accuracy was observed at epoch 9, where the training accuracy was 0.8957, the validation accuracy was 0.8707, the training loss was 0.2692, and the validation loss was 0.3241.

After epoch 9, the validation loss stopped decreasing, and the validation accuracy started to decline, even though the training accuracy continued to increase. This indicates that the model was overfitting to the training data. A similar trend can be observed in the loss curve in Figure 5, where the validation loss initially decreased but started increasing after epoch 9.

As a result, training was stopped at epoch 14 using early stopping to prevent further overfitting. This ensured that the model retained its best validation accuracy while avoiding unnecessary additional training that could degrade generalization performance.

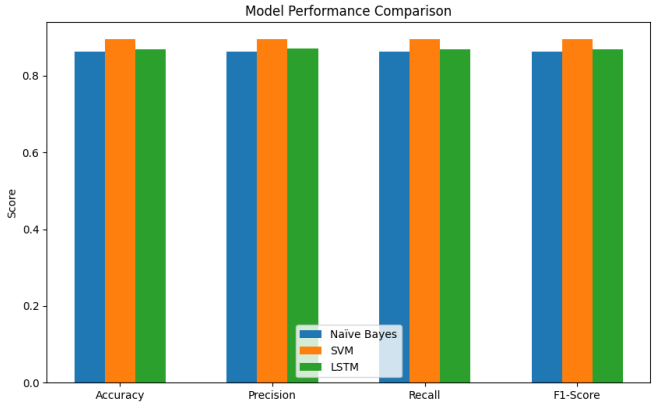

Fig. 6: Comparison of accuracy, precision, recall, and F1-score for Naïve Bayes, SVM, and LSTM.

A more detailed comparison can be seen in Figure 6, where additional performance metrics were given. SVM outperformed both Naïve Bayes and LSTM across all metrics. The LSTM model, despite its strong performance, showed slightly lower precision and recall compared to SVM. Naïve Bayes, while performing well, was consistently the weakest among the three models. These results indicate that SVM was the most effective approach for sentiment classification in this study.

Another useful way to evaluate the models is through the Receiver Operating Characteristic (ROC) curve, which illustrates the trade-off between the true positive rate and false positive rate at various classification thresholds. Figure 7 presents the ROC curves and the corresponding Area Under the Curve (AUC) scores for each model.

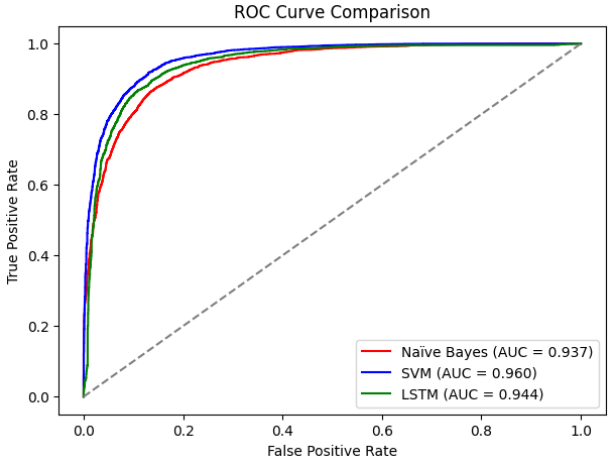

Fig. 7: ROC curve comparison for Naïve Bayes, SVM, and LSTM.

As seen in Figure 7, SVM achieved the highest AUC score, followed closely by LSTM, with Naïve Bayes having the lowest performance. This ranking aligns with the test accuracy results, further reinforcing the effectiveness of SVM in sentiment classification.

Lastly, Figure 8 presents the confusion matrices for all three models.

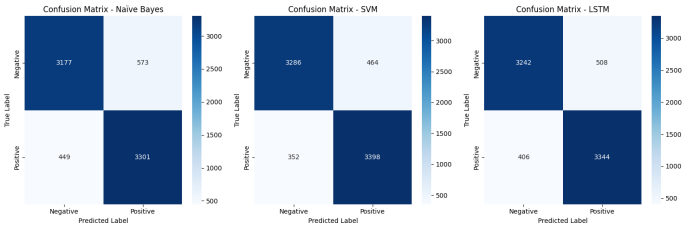

Fig. 8: Confusion matrices for Naïve Bayes, SVM, and LSTM models.

Confusion matrices confirm that SVM performed the best in all aspects, achieving the highest number of correctly classified instances and the lowest number of false positives and false negatives.

The differences in model performance can be attributed to their underlying mechanisms. SVM achieved the highest accuracy due to its ability to find an optimal decision boundary in high-dimensional space, making it more effective in handling text features. Naïve Bayes performed well but was limited by its assumption of feature independence, which restricted its ability to capture complex word relationships. The LSTM model, despite leveraging sequential dependencies, did not surpass SVM, likely due to the need for larger datasets to fully utilize its learning potential.

V. Conclusion

This project aimed to classify IMDB movie reviews as positive or negative by applying machine learning and deep learning techniques on textual data. Three models were trained: Naïve Bayes, Support Vector Machine (SVM), and Long Short-Term Memory (LSTM). All models surpassed the target accuracy of 0.8, demonstrating their effectiveness in sentiment classification, with SVM achieving the highest accuracy of 0.8912.

Several challenges were encountered during the study. The LSTM model exhibited overfitting, as seen in the learning curves, highlighting the need for a larger dataset or regularization techniques to improve generalization. Naïve Bayes, while simple and efficient, struggled with capturing contextual dependencies due to its assumption of feature independence. SVM proved to be the most robust, benefiting from its margin-maximizing property and ability to handle high-dimensional text features.

Future improvements could involve integrating state-of-the-art transformer-based models such as BERT, which leverage bidirectional attention mechanisms to capture deeper contextual relationships. Additionally, hyperparameter optimization and data augmentation techniques could further enhance model performance and generalization.

VI. Accessing the Project

The work for this project can be found in notebook. To reproduce the results, simply running the program is sufficient.