**DLBAIPNLP01 - Project: NLP**

# PROJECT REPORT

**Task 1:**

Sentiment Analysis on Movie Reviews


**Course of Study:**

Bachelor of Science Applied Artificial Intelligence

**Tutor Name:**

Dr. Aditya Mushyam

**Student Name:**

Achshah R M

**Matriculation No.:**

92125572


**Submission Date: 08/07/2024**

## Table of Contents

## List of Figure

# 1. Introduction

With the proliferation of social media, movie reviews have become a significant source of public opinion about films. These reviews offer valuable insights for movie studios to gauge the general sentiment towards their productions. Understanding this sentiment is crucial for making informed decisions about marketing strategies, production choices, and audience engagement (Pang & Lee, 2008). Sentiment analysis, a subset of Natural Language Processing (NLP), involves the systematic identification and categorization of opinions expressed in text, particularly to determine the writer's attitude towards a specific subject or overall contextual polarity of the sentiment (Liu, 2012).

The primary objective of this project is to develop an NLP system capable of performing sentiment analysis on movie reviews. The system will classify each review as either positive or negative, providing an overall sentiment towards the movie. This binary classification will help movie studios and related stakeholders to understand audience reactions effectively.

To achieve the objective, this project encompasses several key tasks. It begins with data collection, where movie reviews are gathered from publicly available datasets, such as Stanford's Large Movie Review Dataset (Maas et al., 2011). The next step is data preprocessing, which involves cleaning the collected data, tokenizing the text, removing stop words, and applying lemmatization to standardize the text. Following this, the model training phase utilizes supervised learning algorithms, including Naive Bayes, Random Forest, Support Vector Machines (SVM), and Long Short-Term Memory (LSTM) models, to train the sentiment analysis model. The performance of the trained model is then assessed using a validation set to evaluate and optimize the model, followed by a final evaluation on a separate test dataset to determine its accuracy and generalization capabilities. To demonstrate the practical application of the model, a simple application is developed to analyse movie reviews.

This report details the step-by-step approach taken to develop the sentiment analysis system. It includes the methods used for data preprocessing, the rationale behind the choice of machine learning algorithms, the experimental setup, and the evaluation metrics. Additionally, the report discusses the results obtained from various experimental setups and provides insights into the system's performance and potential areas for improvement. By following a structured approach, this project aims to contribute to the field of NLP by providing a practical solution for sentiment analysis in the context of movie reviews.

# 2. Methods

## 2.1 Data Collection

The data collection process is critical, as the quality and quantity of data used for training significantly impact the performance and accuracy of machine learning algorithms. As highlighted by Ng (2018), "the success of any machine learning project hinges largely on the quality of the data used" (p. 45).

Therefore, obtaining high-quality and substantial data is essential for developing an effective sentiment analysis model.

There are various methods for collecting data, including web scraping and using publicly available datasets. Web scraping involves extracting data from websites, which can be time-consuming and may require significant effort to ensure data quality and labelling. On the other hand, publicly available datasets are often pre-processed and labeled, saving time and ensuring data quality and consistency.

For this project, we opted to use a high-quality publicly available dataset, specifically the sentiment analysis dataset provided by Maas et al. (2011). This dataset was chosen due to its extensive collection of 50,000 labeled movie reviews, which are evenly split between positive and negative sentiments. This pre-labeled dataset not only reduced the time required for data preparation but also provided a robust foundation for training and evaluating our sentiment analysis model.

### 2.2 Data Preprocessing

Data preprocessing is a crucial step in preparing raw data for machine learning models, as it significantly impacts the model's performance and accuracy. According to Kotsiantis et al. (2006), "data preprocessing techniques are essential for improving the quality of data and consequently the quality of the results obtained from machine learning algorithms" (p. 4).

#### *2.2.1 Data Cleaning*

The dataset used for this project contained HTML tags and special characters, which needed to be removed to ensure that the text data is clean and consistent. Removing these elements helps in reducing noise and focusing on the actual content of the reviews, thereby improving the model's ability to learn meaningful patterns. For this purpose, BeautifulSoup, a Python library for parsing HTML and XML documents, and the re module, which provides support for regular expressions, were chosen. BeautifulSoup efficiently extracts text from HTML, while the re module allows for precise removal of unwanted characters, making them a robust combination for data cleaning tasks (Richardson, 2013).

#### *2.2.2 Tokenization*

Tokenization is the process of splitting text into individual words or tokens, which is essential for converting raw text into a format that can be processed by machine learning algorithms. This step is necessary because most NLP algorithms operate on individual words or tokens rather than entire sentences or paragraphs. Punkt, a pre-trained model for tokenization available in the NLTK library, was used for this task. Punkt is advantageous due to its ability to handle various linguistic nuances and punctuation marks effectively, making it suitable for tokenizing complex texts (Kiss & Strunk, 2006).

### *2.2.3 Stop Word Removal*

Stop word removal involves eliminating common words that do not contribute significant meaning to the text, such as "and," "the," and "is." This step is crucial for reducing the dimensionality of the data and focusing on the words that carry meaningful information. The NLTK's stopwords model was used for this purpose. This model is advantageous as it provides a comprehensive list of stopwords for multiple languages and can be easily customized to include additional words if needed (Bird, Klein, & Loper, 2009).

### *2.2.4 Lemmatization*

Lemmatization reduces words to their base or root form, which helps in standardizing the text data and improving the consistency of word representations. For this project, the WordNet lemmatizer and the omw-1.4 models were used. Unlike stemming, which simply trims words to a base form, lemmatization considers the context and morphological analysis of words, making it more accurate and effective for preserving the meaning of words (Miller, 1995).

### *2.2.5 Encoding*

Encoding transforms text data into numerical representations that machine learning models can process. For this project, TF-IDF (Term Frequency-Inverse Document Frequency) transformation was used for models like SVM, Random Forest, and Naive Bayes. TF-IDF is effective for capturing the importance of words in a document relative to a corpus, making it suitable for text classification tasks (Ramos, 2003). For the LSTM model from TensorFlow, tokenization and padding were employed. Tokenization converts text into sequences of integers, while padding ensures that all sequences have the same length, which is essential for batch processing in neural networks (Chollet, 2018).

## 2.3 Choice of algorithms

Choosing the right algorithm for an NLP task is crucial as it directly impacts the model's ability to capture patterns and make accurate predictions (Jain et al, p. 35, 2009). For this project, we considered a range of algorithms: Naive Bayes, Random Forest, Support Vector Machines (SVM), and Long Short-Term Memory (LSTM) networks.

### *2.3.1 Naïve Bayes*

Naïve Bayes is a popular baseline model for NLP tasks due to its simplicity and effectiveness. It is a probabilistic classifier based on Bayes' theorem, which assumes that the features are conditionally independent given the class label. Despite its simplicity, Naïve Bayes performs surprisingly well on many NLP tasks, making it a good starting point for text classification (Manning, Raghavan, & Schütze, 2008). The algorithm calculates the probability of each class given the input features and selects the class with the highest probability. Its advantages include ease of implementation, fast training, and low computational cost.

### *2.3.2 Random Forest*

Random Forest is a powerful ensemble method that combines multiple weak estimators to form a strong estimator. It works by constructing a multitude of decision trees during training and outputting the class that is the mode of the classes (classification) of the individual trees. This method reduces overfitting and improves generalization by averaging multiple trees' results (Breiman, 2001). Random Forest is particularly useful for handling large datasets with high dimensionality, making it a suitable choice for text classification tasks.

### *2.3.3 Support Vector Machines*

Support Vector Machines (SVM) works by finding the hyperplane that best separates the data into different classes. It maximizes the margin between the closest points of the classes, known as support vectors, ensuring robustness against overfitting (Cortes & Vapnik, 1995). SVMs are well-suited for high-dimensional spaces and have been shown to perform well in text classification tasks due to their ability to handle sparse data effectively.

### *2.3.4 Long Short-Term Memory (LSTM)*

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) that are particularly effective for tasks involving sequential data, such as text. LSTMs are designed to capture long-term dependencies by using special units called memory cells to maintain information over long periods (Hochreiter & Schmidhuber, 1997). This capability makes LSTMs well-suited for sentiment analysis, where the context of words within a sentence is crucial for understanding sentiment. LSTMs have been proven to perform well in various NLP tasks, including sentiment analysis, due to their ability to model complex sequences and long-range dependencies.

## 2.4 Evaluation Metrics

Evaluating the performance of machine learning models using statistical and mathematical methods is crucial for understanding their effectiveness and suitability for specific use cases (Powers, p. 37, 2011). For this project, we used several evaluation metrics, including accuracy score, precision, recall, F1 score, and the ROC curve with AUC score, to comprehensively assess the performance of our sentiment analysis models.

### *2.4.1 Accuracy Score*

The accuracy score provides an overall indication of how well the model performs across all classes. In the context of sentiment analysis, a high accuracy score indicates that the model correctly identifies positive and negative sentiments a majority of the time. However, accuracy alone can be misleading, especially with imbalanced datasets, which is why it is often used in conjunction with other metrics (Sokolova & Lapalme, 2009).

### *2.4.2 Precision*

Precision measures the accuracy of the positive predictions made by the model. In sentiment analysis, high precision means that when the model predicts a review as positive, it is likely to be correct. This is particularly important when the cost of false positives is high, as it helps ensure that positive classifications are reliable (Powers, 2011).

### *2.4.3 Recall*

Recall, also known as sensitivity, measures the model's ability to identify all relevant instances. In sentiment analysis, high recall indicates that the model successfully captures a large portion of the actual positive reviews, which is crucial when it is important to identify all positive cases, even at the risk of including some false positives (Saito & Rehmsmeier, 2015).

### *2.4.4 F1 Score*

The F1 score is the harmonic mean of precision and recall, providing a balance between the two metrics. The F1 score is particularly useful when the dataset is imbalanced, as it takes both false positives and false negatives into account. In sentiment analysis, a high F1 score indicates that the model maintains a good balance between precision and recall, providing a more comprehensive evaluation of the model's performance (Chicco & Jurman, 2020).

### *2.4.5 ROC Curve with AUC Score*

The ROC (Receiver Operating Characteristic) curve is a graphical representation of a model's diagnostic ability, plotting the true positive rate (recall) against the false positive rate. The AUC (Area Under the Curve) score quantifies the overall ability of the model to discriminate between positive and negative classes. A higher AUC score indicates better performance. The ROC curve and AUC score are valuable for evaluating model performance, particularly in binary classification tasks like sentiment analysis, as they provide insights into the trade-offs between sensitivity and specificity (Bradley, 1997).

## 2.5 Implementation

To demonstrate the practical application of the developed sentiment analysis model, a simple web application was created. This application allows users to input movie reviews and receive an output indicating the sentiment as positive or negative. The web application was built using Flask, object-oriented Python programming (OOP), and HTML.

### *2.5.1 Flask*

Flask is a micro web framework for Python that is lightweight and easy to use, making it an ideal choice for developing small to medium-sized web applications. Flask provides the essential tools needed to build a web application, such as routing, request handling, and templating, without the overhead of a full-stack framework like Django. Flask's simplicity and flexibility allow for rapid development and easy integration with machine learning models (Grinberg, 2018). The choice of

Flask over Django was driven by its minimalistic nature, which is well-suited for a project of this scope, where the primary focus is on showcasing the sentiment analysis model rather than building a complex web application (Armin, 2020).

#### *2.5.2 Object-Oriented Python Programming (OOP)*

Object-oriented programming (OOP) is a programming paradigm that uses objects and classes to structure code. OOP was chosen for this project due to its ability to encapsulate data and functions, promote code reusability, and enhance maintainability. Using OOP allows for a clear and modular design, where different components of the application, such as data preprocessing, model prediction, and web handling, can be developed independently and integrated seamlessly (Lutz, 2013). This approach contrasts with simple scripting, which can become unwieldy and difficult to manage as the complexity of the application grows (Al Sweigart, 2019).

#### *2.5.3 HTML*

HTML (HyperText Markup Language) is the standard language for creating web pages. It provides the structure of the web application, enabling the presentation of text, images, and forms. HTML was used to create the user interface of the web application, allowing users to input their movie reviews and view the sentiment analysis results. The use of HTML ensures that the application is accessible via any web browser, providing a user-friendly and interactive platform for demonstrating the sentiment analysis model (Duckett, 2011).

## 3. Experimental Set-up

### 3.1 Data

The dataset for this project contains 50,000 movie reviews, equally divided between positive and negative sentiments, making it ideal for binary classification tasks. Notably, the dataset has no missing values, ensuring completeness, and includes a few duplicate entries (418 duplicate entries), which are retained as they do not significantly affect model performance. The dataset is balanced, preventing class bias and ensuring equal learning from both positive and negative examples, as confirmed by the label distribution bar chart in figure 1. Reviews contain HTML tags and special characters, necessitating removal during preprocessing to clean the text for model training. The review length distribution is left-skewed, indicating most reviews are short with some long outliers, as shown in a text length distribution graph in figure 2. Common terms in positive reviews include "love," "great," "good," "film," and "one," while negative reviews frequently feature words like "bad," "movie," and "script," visualized in respective word clouds in figure 3. These characteristics guide the preprocessing and training stages, ensuring effective sentiment analysis.

### 3.2 LSTM Architecture

The LSTM model architecture used for this project was designed to effectively capture sequential patterns in the text data for sentiment analysis. The model consisted of an Embedding layer that

transforms input data into dense vectors of size 128. This was followed by a Bidirectional LSTM layer with 64 units, incorporating dropout and recurrent dropout rates of 0.2 to prevent overfitting, and set to return sequences for the next LSTM layer. Another Bidirectional LSTM layer with 32 units was added, also with dropout and recurrent dropout rates of 0.2, but without returning sequences. Finally, a Dense layer with a single neuron and sigmoid activation function provided the probability of the input text being positive. The model was compiled using the binary cross-entropy loss function and the Adam optimizer with a learning rate of 0.001. The model was trained using the data and performance was evaluated. The architecture of the LSTM model is represented in Appendix A.

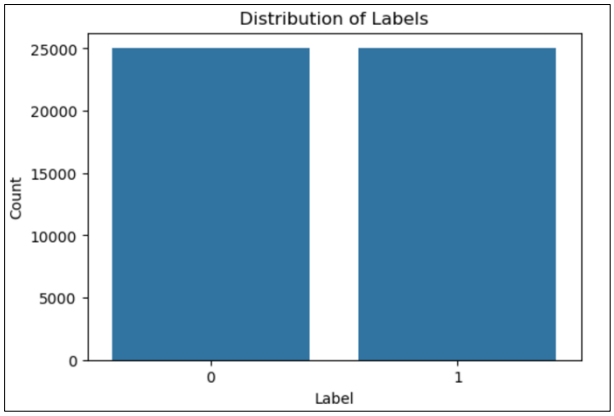



Figure 1. Distribution of Labels

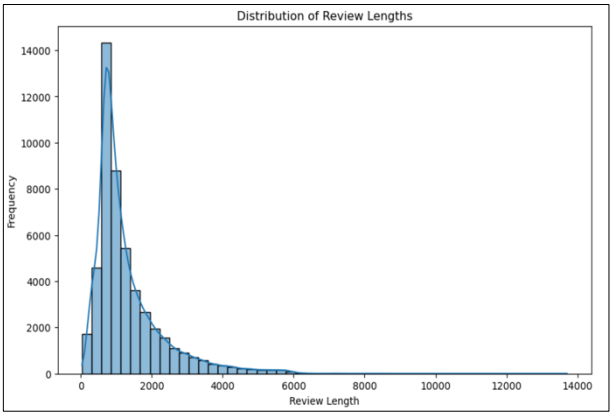



Figure 2. Distribution of Review Lengths


Figure 3. Word Cloud of Positive and Negative Reviews

### 3.3 Web Application System

The web application was developed to enable users to input movie reviews and receive sentiment analysis results using the optimized LSTM model. This application was built using Flask, object-oriented Python programming (OOP), and HTML, each playing a critical role in the system.

The flow of the application begins with the user entering a movie review on the web page. Upon submission, the review is processed by the /analyze route in the Flask application. The submitted review is then passed to the analyze_review function, which handles the preprocessing and sentiment analysis using the LSTM model. The resulting sentiment (positive or negative) is then displayed back on the web page. The component interaction diagram of the web application is represented in Appendix B.

The main components of the web application include app.py, analyse.py, model.py, and preprocess.py. The app.py file defines the Flask application, setting up routes to render the homepage and handle form submissions. The analyse.py file contains the analyze_review function, which processes the review, cleans the text, tokenizes it, pads the sequences, and uses the LSTM model for prediction. The model.py script loads the trained LSTM model and tokenizer, crucial for converting text data into a format suitable for model input. The preprocess.py script includes the clean_review function, which removes HTML tags, special characters, and stop words, and performs lemmatization to prepare the text data for analysis. This well-integrated system effectively demonstrates the practical application of the optimized LSTM model in sentiment analysis, providing accurate and reliable results for movie reviews.

### 3.4 Evaluation Process

The data was split into three sets: 80% for training, 10% for validation, and 10% for testing. All four selected models (Naive Bayes, Random Forest, SVM, and LSTM) were trained on the training data. Due to SVM's limitations in handling large datasets, a sample subset of the training data was used specifically for SVM training. The models were then evaluated on the validation set, and their performances were compared.

The best two models, based on validation set performance, were chosen for further optimization. The validation set was used to fine-tune the model parameters of these selected models, ensuring they performed optimally. The final evaluation of the optimized models was conducted on the test set to assess their performance on new, unseen data, ensuring the models were not biased and could generalize well.

The best-performing model from this process was selected for the development of the web application and was further tested under different scenarios to ensure robustness. The final LSTM model was evaluated across 15 different review scenarios, with the model's responses recorded to assess its effectiveness in various contexts.

## 4. Result and Discussion

The performance of the sentiment analysis models was rigorously evaluated using both default parameters and further optimization processes to identify the best-performing model for deployment in a web application.

### 4.1 Initial Training and Validation Results

The initial training and validation of the sentiment analysis models yielded insightful results. The Naive Bayes model achieved a validation accuracy of 0.8544, with a precision of 0.86, recall of 0.85, and F1-score of 0.86 for class 0 (negative), and a precision of 0.84, recall of 0.86, and F1-score of 0.85 for class 1 (positive). These results indicate that the Naive Bayes model performed well in balancing precision and recall, making it a robust baseline model.

The Random Forest model, on the other hand, showed a slightly lower validation accuracy of 0.8430. The precision, recall, and F1-score for both classes were consistent at 0.84, suggesting that while the model performed adequately, it did not excel in distinguishing between positive and negative sentiments as effectively as the Naive Bayes model.

The SVM model demonstrated a validation accuracy of 0.8552, with class 0 (negative) achieving a precision of 0.88, recall of 0.83, and F1-score of 0.85, and class 1 (positive) achieving a precision of 0.83, recall of 0.88, and F1-score of 0.86. This indicates that the SVM model was particularly strong in maintaining high precision and recall, making it a competitive option for sentiment analysis tasks.

The LSTM model outperformed the other models with a validation accuracy of 0.8590. However, during the training over 10 epochs, the model showed initial improvement in accuracy but started to exhibit signs of overfitting after a few epochs.

The performance of all four models was rigorously compared to determine the most suitable candidates for further optimization, shown in figure 4. Given these results, both the SVM and LSTM models were selected for optimization due to their superior performance.

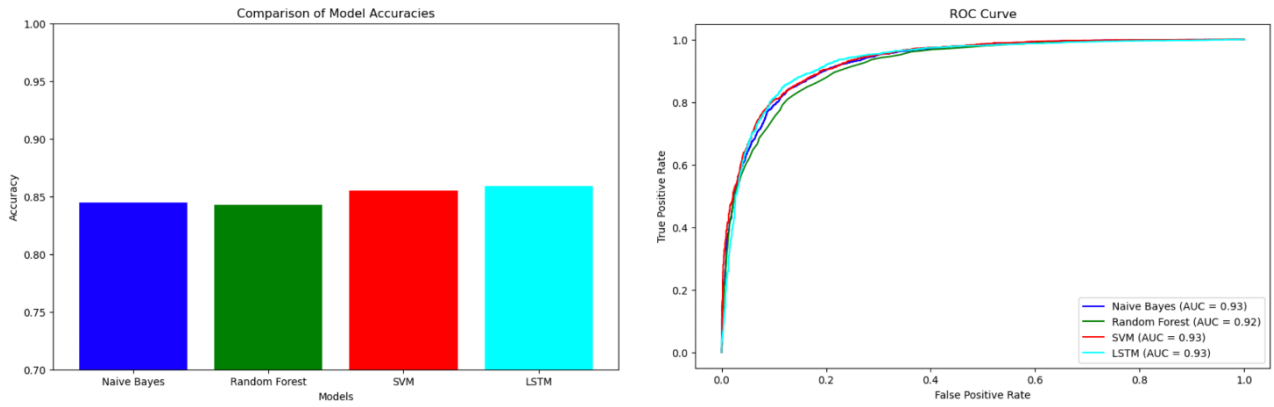



Figure 4. Performance Comparison of Models Using Accuracy Score and ROC Curve

### 4.2 Model Optimization and Test Result

During optimization, the model was enhanced by increasing the dropout and recurrent dropout rates to 0.5 in both Bidirectional LSTM layers to further reduce overfitting. Additionally, early stopping was implemented to monitor the validation loss and restore the best weights if no improvement was seen for three consecutive epochs. The optimized LSTM training performance showed significant improvements by achieving a validation accuracy of 0.8766, indicating improved generalization and reduced overfitting compared to the initial training phase. Figure 5 demonstrates the model accuracy and model loss during training and validation.

In contrast, the SVM model was optimized using Random Search, a technique to find the best hyperparameters by randomly sampling the parameter space. Despite these efforts, the SVM model showed no significant change in its performance, maintaining a validation accuracy of 0.8552. This

outcome highlighted the robustness of the SVM model but also indicated that its performance had reached an optimal level within the constraints of the given parameter space.

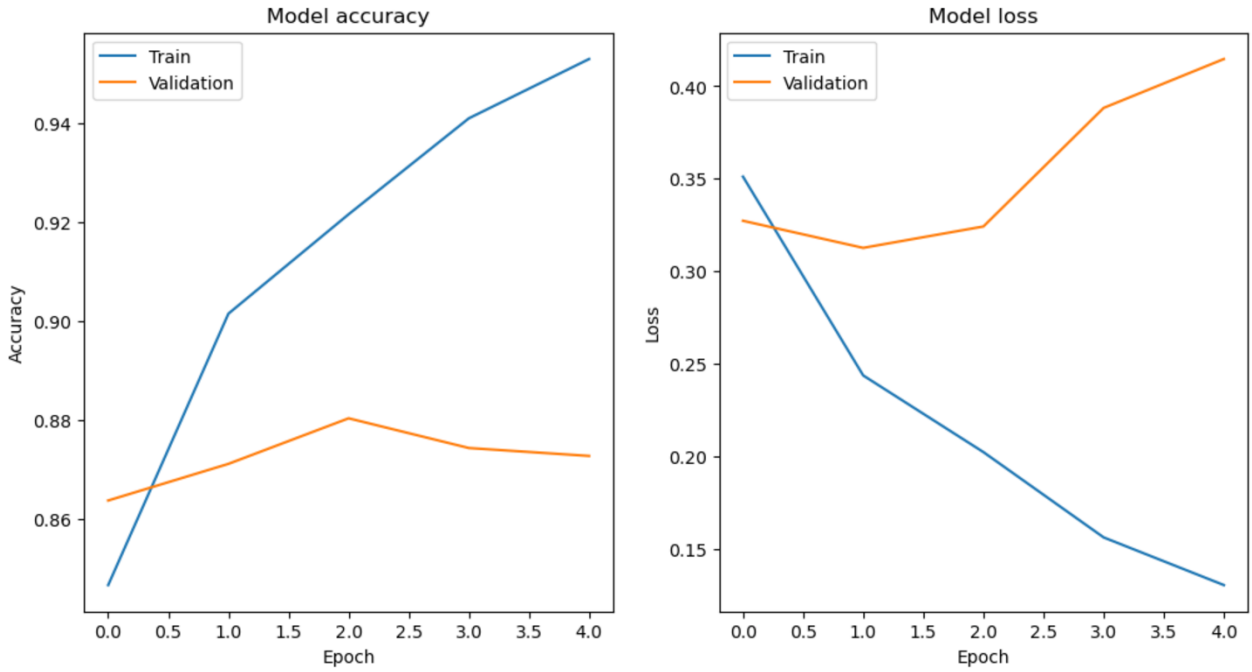



Figure 5. LSTM Model Accuracy and Loss Across Epochs

Both optimized models were then evaluated on the test set to ensure their performance translated well to unseen data. The LSTM model demonstrated robust performance with a test accuracy of 0.8766, confirming its ability to generalize effectively beyond the training and validation datasets. Similarly, the SVM model maintained its performance with a test accuracy of 0.8552. These results validated the optimization process and reinforced the decision to use the LSTM model for the web application, given its superior accuracy and improved handling of overfitting.

### 4.3 Scenario-Based Testing

After integrating the LSTM model into the web application, its performance was rigorously tested under 15 different review scenarios to ensure robustness and reliability. The model consistently provided accurate sentiment predictions for various types of reviews, including short and long reviews, mixed sentiment reviews, genre-specific reviews, sarcastic reviews, slang reviews, and mixed language reviews. The detailed results are presented in Appendix C.

In 14 out of the 15 scenarios, the LSTM model performed exceptionally well, accurately predicting the sentiment of the reviews. This demonstrated the model's ability to handle a wide range of inputs and contexts effectively, making it a reliable tool for sentiment analysis in practical applications.

However, the model struggled to correctly predict the sentiment of mixed language reviews (shown in figure 6), suggesting that the training data did not adequately cover such cases. This shortfall highlights an area for future improvement, where incorporating mixed language data into the training process could significantly enhance the model's performance and broaden its applicability.

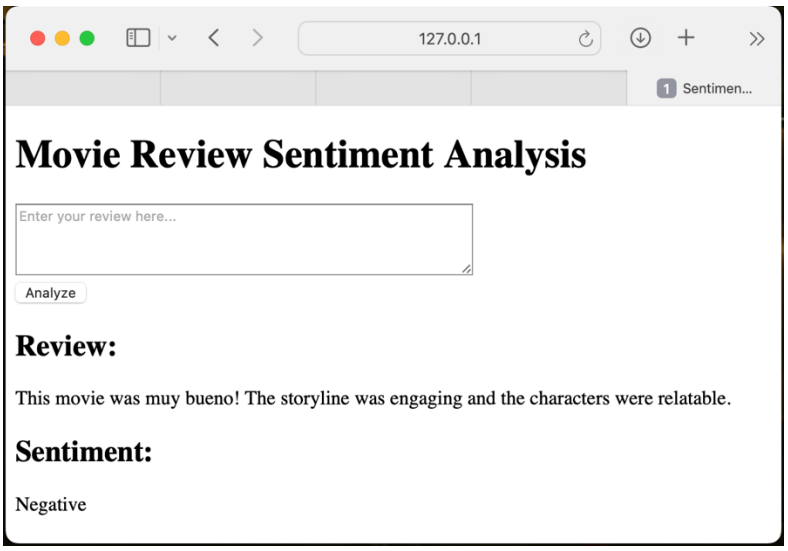



Figure 6. LSTM Model Prediction of Mixed Language Review

## 5. Conclusion

The project successfully met its primary objective of developing an effective sentiment analysis system. The LSTM model, integrated into a web application, proved to be a reliable tool for analysing movie reviews. The careful selection and optimization of the LSTM model, along with comprehensive evaluation, ensured that the system was both accurate and robust. The project not only demonstrated the application of theoretical concepts in a practical setting but also highlighted the importance of model evaluation and optimization in developing reliable NLP systems.

The findings from this project have significant implications for future research and practical applications. The successful deployment of the LSTM model in a web application underscores the potential of deep learning models in real-world sentiment analysis tasks. This project also emphasizes the need for thorough evaluation and optimization to address issues such as overfitting. The insights gained from scenario-based testing can inform the development of more resilient models capable of handling diverse and complex inputs.

While the project achieved its objectives, several areas for future improvement were identified. Incorporating mixed language data into the training process could enhance the model's ability to handle multilingual reviews. Additionally, exploring advanced techniques such as transfer learning and domain adaptation could further improve the model's performance and applicability across different domains. Expanding the dataset to include more varied and nuanced reviews can also contribute to building a more comprehensive and robust sentiment analysis system.

This project has laid a solid foundation for developing effective sentiment analysis systems using deep learning models. The success of the LSTM model in various testing scenarios highlights its potential for broader applications. Future research should focus on addressing the identified limitations and exploring advanced techniques to further enhance the model's performance and adaptability.

**Link to Project:** https://github.com/Achshah-RM/-Sentiment-Analysis-on-Movie-Reviews/

## Bibliography


1. Liu, B. (2022). *Sentiment analysis and opinion mining* (1st ed.). Springer Cham. https://doi.org/10.1007/978-3-031-02145-9
2. Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies* (pp. 142-150). Association for Computational Linguistics. http://www.aclweb.org/anthology/P11-1015
3. Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval, 2*(1-2), 1-135. http://dx.doi.org/10.1561/1500000011
4. Ng, A. (2018). *Machine Learning Yearning*. deeplearning.ai. https://nessie.ilab.sztaki.hu/~kornai/2020/AdvancedMachineLearning/Ng_MachineLearningYearning.pdf
5. Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with Python*. O'Reilly Media, Inc. https://www.oreilly.com/library/view/natural-language-processing/9780596803346/
6. Chollet, F. (2018). *Deep learning with Python*. Manning Publications.

   https://www.manning.com/books/deep-learning-with-python
7. Kiss, T., & Strunk, J. (2006). Unsupervised multilingual sentence boundary detection. *Computational Linguistics, 32*(4), 485-525. https://doi.org/10.1162/coli.2006.32.4.485
8. Kotsiantis, S., Zaharakis, I. D., & Pintelas, P. E. (2006). Machine learning: A review of classification and combining techniques. *Artificial Intelligence Review, 26*(3), 159-190. https://doi.org/10.1007/s10462-007-9052-3
9. Miller, G. A. (1995). WordNet: a lexical database for English. *Communications of the ACM, 38*(11), 39-41. https://doi.org/10.1145/219717.219748
10. Ramos, J. (2003). Using TF-IDF to determine word relevance in document queries. In *Proceedings of the First Instructional Conference on Machine Learning*. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=b3bf6373ff41a115197cb5b30e57830c16130c2c
11. Richardson, L. (2013). *Beautiful Soup documentation*. Crummy.

    https://www.crummy.com/software/BeautifulSoup/bs4/doc/
12. Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5-32. https://doi.org/10.1023/A:1010933404324

13. Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning, 20*(3), 273-297. https://doi.org/10.1007/BF00994018

14. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

15. Jain, A. K., Duin, R. P. W., & Mao, J. (2000). Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 22*(1), 4-37. https://doi.org/10.1109/34.824819

16. Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press. https://doi.org/10.1017/CBO9780511809071

17. Bradley, A. P. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition, 30*(7), 1145-1159. https://doi.org/10.1016/S0031-3203(96)00142-2

18. Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics, 21*(1), 6. https://doi.org/10.1186/s12864-019-6413-7

19. Powers, D. M. (2011). Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *Journal of Machine Learning Technologies, 2*(1), 37-63. https://doi.org/10.48550/arXiv.2010.16061

20. Saito, T., & Rehmsmeier, M. (2015). The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLoS ONE, 10*(3), e0118432. https://doi.org/10.1371/journal.pone.0118432

21. Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management, 45*(4), 427-437. https://doi.org/10.1016/j.ipm.2009.03.002

22. Armin, R. (2020). *Flask web development: Developing web applications with Python*. O'Reilly Media.

23. Duckett, J. (2011). *HTML & CSS: Design and build web sites*. Wiley.

24. Grinberg, M. (2018). *Flask web development: Developing web applications with Python*. O'Reilly Media.

25. Lutz, M. (2013). *Learning Python*. O'Reilly Media.

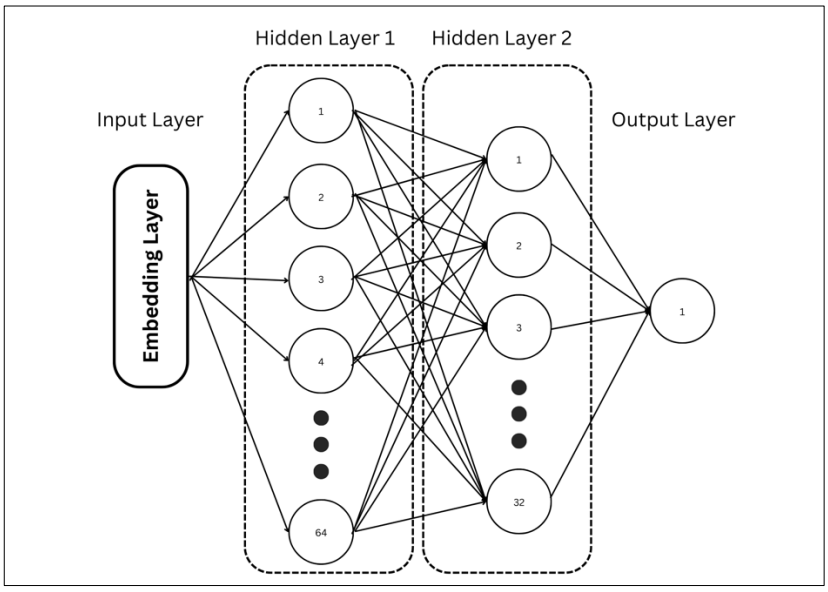
26. Sweigart, A. (2019). *Automate the boring stuff with Python: Practical programming for total beginners*. No Starch Press.

## List of Appendices

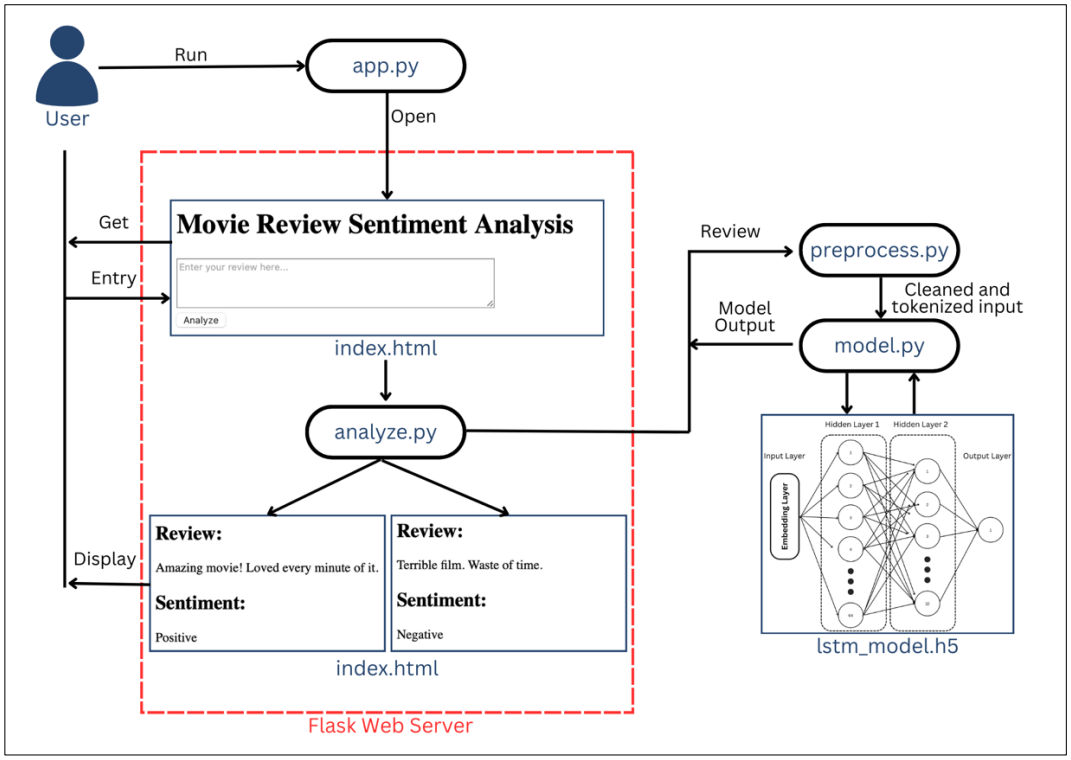


## Appendices

Appendix A: Architecture of the LSTM Model




Appendix B: Component Interaction Diagram of Web Application System



Appendix C: Scenario-Based Experiment Results

| Review Type | Review | Expected Sentiment | Model Output (LSTM) |
|---|---|---|---|
| Short Positive Review | "Amazing movie! Loved every minute of it." | Positive | Positive |
| Short Negative Review | "Terrible film. Waste of time." | Negative | Negative |
| Long Positive Review | "This movie was fantastic from start to finish. The acting was superb, the storyline was gripping, and the cinematography was stunning. I couldn't take my eyes off the screen. Highly recommended for anyone who loves a good drama." | Positive | Positive |
| Long Negative Review | "I was extremely disappointed with this movie. The plot was all over the place, the characters were underdeveloped, and the pacing was painfully slow. I struggled to stay awake through the entire film. Definitely not worth watching." | Negative | Negative |
| Positive Review with Mixed Sentiments | "The movie started off a bit slow, but once it picked up, it was incredible. The acting was top-notch and the storyline was very engaging." | Positive | Positive |
| Negative Review with Mixed Sentiments | "While the acting was decent, the storyline was too predictable and lacked originality. The movie failed to keep my interest." | Negative | Negative |
| Genre-Specific Positive Review (Comedy) | "Hilarious movie! The jokes were on point and I laughed throughout the entire film." | Positive | Positive |
| Genre-Specific Negative Review (Comedy) | "The humor was forced and fell flat. I didn't find the movie funny at all." | Negative | Negative |
| Genre-Specific Positive Review (Horror) | "Scary and thrilling! This horror movie kept me on the edge of my seat the whole time." | Positive | Positive |
| Genre-Specific Negative Review (Horror) | "Not scary at all. The movie was filled with clichés and predictable jump scares." | Negative | Negative |
| Genre-Specific Positive Review (Action) | "Non-stop action! The fight scenes were amazing and the stunts were incredible." | Positive | Positive |
| Genre-Specific Negative Review (Action) | "The action scenes were overdone and the plot was weak. I couldn't stay interested." | Negative | Negative |
| Review with Sarcasm | "Oh great, another movie with a predictable plot and wooden acting. Just what I needed." | Negative | Negative |
| Review with Slang | "This movie was lit! The actors did an awesome job and the plot was dope." | Positive | Positive |
| Mixed Language Review | "This movie was muy bueno! The storyline was engaging and the characters were relatable." | Positive | Negative |