Natural Language Computing

3CS102DC24

Innovative Assignment

"Building a Smart Feedback System: Integrating Sentiment Analysis and Content Moderation"

B. Tech. Semester V

Prepared By

(PATEL MONISH)
(23BCE239)

(PATEL PRAPTI)

(23BCE241)

(PATEL PRIYANSH)
(23BCE243)

(RAJSHREE CHAVDA)

(24BCE508)

School of Engineering
Institute of Technology
Nirma University
Ahmedabad 382481

1. Introduction

1.1 About Project:

Within today's digital landscape, online feedback and reviews are central to shaping public sentiment of products, services, and organizations, although the ability to efficiently and objectively analyse large volumes of text can prove challenging. Our project, - "Building a Smart Feedback System: Using Sentiment Analysis and Content Moderation", responds to this gap through a data-driven approach that utilizes machine learning models to automate the interpretation of user opinion, as well as responsible content.

The main innovation of this system is the comparative analysis of multiple sentiment analysis models to determine the optimal model that provides the most accurate and efficient analysis of feedback. Incorporating models into a cohesive framework will allow for real-time analysis of feedback while providing a comparison of model performance capable of being integrated for an intelligent adaptable feedback management tool.

1.2 Goal of Project:

This project's main objective is to evaluate and compare various sentiment analysis algorithms to find the best fit for actual feedback systems. The project provides a performance evaluation of all classifiers, rather than selecting just one. This study is an opportunity to not only see which model is best, but also allows you to understand how models behave with different kinds of textual data. Furthermore, as an additional feature, a layer of content moderation has been added to ensure that the feedback remains clean, civil and appropriate.

1.3 Expected Outcomes:

> Identification of the most accurate model for sentiment classification on given feedback data.

> Real-time prediction of sentiment using the best-performing model.

>Filtered and moderated textual feedback that aligns with ethical and communication standards.

> Visual and interpretable comparison results between multiple ML algorithms.

1.4 Scope of Project:

This project's purpose is not limited to sentiment prediction. Instead, it seeks to develop a general feedback analysis system that can be configured for different purposes, such as education, e-commerce, or service organizations. The implementation of several different machine learning models to compare performance, combined with a content moderation capability, enables the system to provide a polarity sentiment, plus ensure that the user-provided information is relevant, respectful, and useful for engaging insights.

The system is modular by design; any advanced deep learning models or large language models can be introduced in the future without changing the overall structure of the sentiment model.

2. Dataset and Resources

2.1 Dataset Overview:

The project utilizes the IMDb Movie Review Dataset available on Kaggle, a widely recognized benchmark dataset for sentiment analysis.
It consists of 50,000 movie reviews, each labelled as either positive or negative, making it ideal for supervised binary text classification tasks.
The dataset is already pre-processed to balance both sentiment classes equally, containing:

- 25,000 positive reviews
- 25,000 negative reviews

This balanced structure ensures that model evaluation metrics like accuracy, precision, and recall are not biased toward any one sentiment class.

2.2 Dataset Characteristics:

Attribute	Description
Source	Kaggle – IMDb Movie Review Dataset
Total Samples	50,000 reviews
Classes	Positive (1), Negative (0)
Average Review Length	~230 words per review
Type	Textual data (unstructured)
Purpose	Sentiment classification (binary)

1.3 Reason to Choose:

The IMDb dataset was chosen due to the following factors:

It is clean, well-labelled, and balanced, so that preprocessing is not overly complicated.

Because the dataset consists of real-world user reviews, it reflects a real-world distribution of sentiment and interesting variation.

Because the dataset is a standard benchmark in sentiment analysis research, the models could be compared fairly and accurately.

It can help more accurately assess how different machine learning algorithms deal with realistic text data, which consists of mixed tone and structural organization.

3. Methodology and System Design

3.1 Data Preprocessing:

Before training, the IMDb dataset undergoes several preprocessing steps to make it suitable for deep learning models.
The main steps include:

- Text cleaning: Removal of HTML tags, punctuation, numbers, and special characters.
- Tokenization: Converting each review into sequences of tokens using Keras or Hugging Face tokenizers.
- Stop word removal and lowercasing: Eliminating common words and standardizing text.
- Padding and truncation: Ensuring all sequences have the same length for model compatibility.
- Embedding preparation: Converting words into dense vector representations using Word2Vec, GloVe, or contextual embeddings generated by BERT.

This process ensures that textual data is consistent, noise-free, and suitable for training all neural models.

3.2 Model Training and Comparison:

The objective of the project is to examine the comparative performance of four deep learning models for sentiment analysis. To do that, each model is trained, evaluated, and visualized through several analytics components in the web interface to differentiate their performance, and each model's efficiencies and performance.

1. Recurrent Neural Network:

The RNN is a basic sequential model that processes an input word by word, but maintains information from the previous steps. An RNN captures word dependencies well in time series settings, like language, while capturing short-term dependencies. However, it does not retain memory for long-term contexts well.

2. Gated Recurrent Unit:

The GRU is a simpler and computationally efficient form of an RNN with gating mechanisms that maintain more context in memory and train/infer faster than RNNs.

3. Long Short-Term Memory:

The LSTM is an advanced form of a recurrent model, designed to "remember" both short-term dependencies and long-term word dependencies. LSTM networks use memory cells to analyze patterns of sentences to understand sentiment more clearly.

4. Bidirectional Encoder Representations from Transformers:

BERT is a transformer-based model that is pre-trained and understands the full bidirectional context of words in a sentence. Overall, BERT is computationally expensive but has the highest accuracy, as depth is achieved through pre-trained deep semantic understanding.

All models have been trained on the same split dataset so comparisons are valid in multiple metrics (accuracy, inference time, and size of the model) for fairness to the model.

3.3 System Architecture:

Backend (Model Layer):

The backend layer is implemented in Python and utilizes TensorFlow or PyTorch. This layer will load all the trained models (RNN, GRU, LSTM, BERT), handle preprocessing the text input, handle inference and produce prediction labels for sentiment polarity.

API Layer:

The API Layer is developed using FastAPI, this layer acts as the interface between the models and the web interface. The API Layer will receive input from the user, process the text input through the selected models, and return prediction results (i.e., labels) and confidence coefficients from the models.

Frontend (User Interface):

The frontend is the user interface built using ReactJS and Tailwind CSS. This allows the user to enter text feedback as well as to receive and view predictions for all four models in real-time.

3.4 Workflow and Real-time Prediction:

The workflow of the system is as follows:

> The user enters a movie review or feedback through the webpage.

> The input is sent to the backend API to pre-process and tokenize the input.

> The pre-processed text is sent sequentially through RNN, GRU, LSTM, and BERT.

> Each model provides a sentiment classification (positive or negative) and a confidence value.

> The backend sends all of the predictions back to the front-end to be displayed for easy visual comparison.

> In summary, this process allows for real-time sentiment analysis; and insight regarding how each model interprets the same input text in a different way.

4. Implementation Details

4.1 Development Environment and Tools:

Python was utilized for the backend model implementation and JavaScript (ReactJS) for the frontend interface in the system that was implemented presented in this work.
All models were trained and evaluated within a deep learning environment that leveraged GPU acceleration for computation. The following lists the key tools and technology used:

> Languages: Python, Javascript
> Deep Learning Libraries: TensorFlow, Keras, and PyTorch (for building and training models)
> Natural Language Processing: NLTK, spaCy, and Hugging Face Transformers
> Backend Framework: FastAPI (for building and integrating machine learning models into application and creating APIs)
> Frontend Framework: React.js with Tailwind.css (for creating visually appealing front-end enriched applications)
> Visualization Libraries: Chart.js, Recharts, and D3.js (for all visualizations comparing algorithms/models)
> Dataset Source: IMDb Movie Review Dataset (Kaggle)
> Development stack: Jupyter Notebooks, VS Code, Postman and GitHub
The aforementioned environment facilitated seamless integration of the machine learning model with the user interface.

4.2 Backend and API Integration:

The backend was built using FastAPI, providing a lightweight and efficient framework for serving model predictions. The workflow of the backend includes:

1. Accepting user input from the frontend through HTTP POST requests.
2. Preprocessing the input text (tokenization, padding, and embedding).
3. Running inference across all models (RNN, GRU, LSTM, BERT).
4. Returning predictions, confidence scores, inference times, and model-specific metrics as structured JSON responses.

FastAPI ensures low latency, asynchronous processing, and easy scalability, making it suitable for handling multiple real-time user requests simultaneously.

4.3 Frontend Development and Visualization:

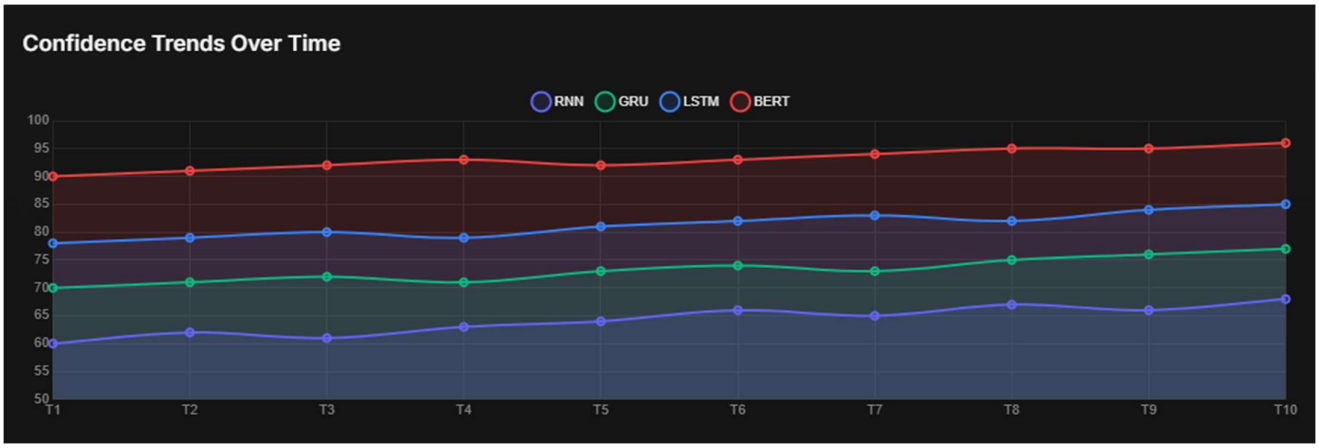
The frontend interface, developed using ReactJS and styled with Tailwind CSS, allows users to interact with the system in an intuitive and visually rich manner.
It provides separate components for each analytical visualization, including:

- Sentiment Comparison Panel to show each model's prediction.
- Confidence Trends to plot model confidence over time.
- Performance Radar and Accuracy Trade-off Charts to visualize model metrics.
- Model Size Distribution and Average Inference Time charts to compare model efficiency.

The frontend communicates with the FastAPI backend using RESTful APIs, dynamically rendering results in real-time based on backend responses.
All data visualizations are implemented using Chart.js and Recharts, providing clear and responsive comparison outputs.

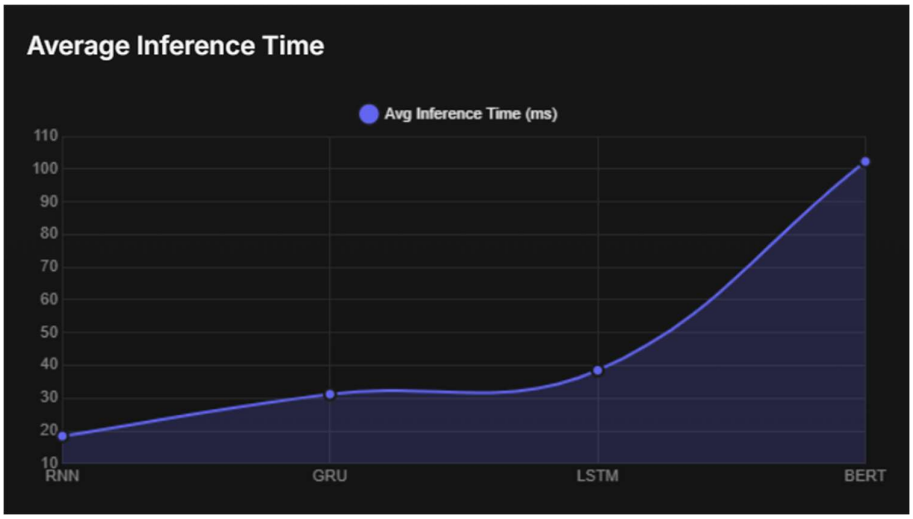
5. Analysis and Discussion

5.1 Confidence Trends Over Time:



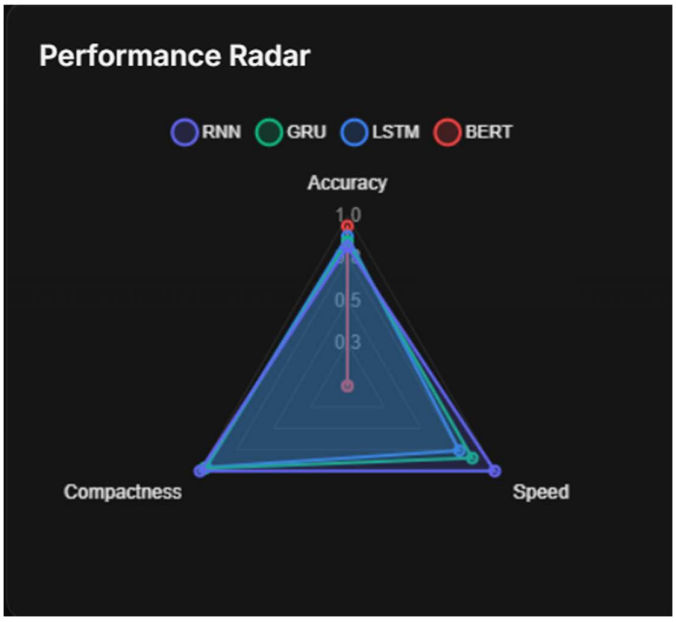
The following figure illustrates the differences in each model's confidence which is defined as how confident each model is about its sentiment prediction over each of the ten evaluations. The higher the confidence level, the higher the confidence that the feedback is classified as positive or negative. In general, models classified with the highest confidence were BERT hovering around 90–95% confidence, LSTM with stable confidence around 80%, GRU around 70% confidence, and RNN at the bottom around 60% confidence. Overall, the trend indicates that BERT as a transformer-based model has stronger confidence and consistent predictions than recurrent models, such as LSTM, GRU, and RNN.

5.2 Average Inference Time:



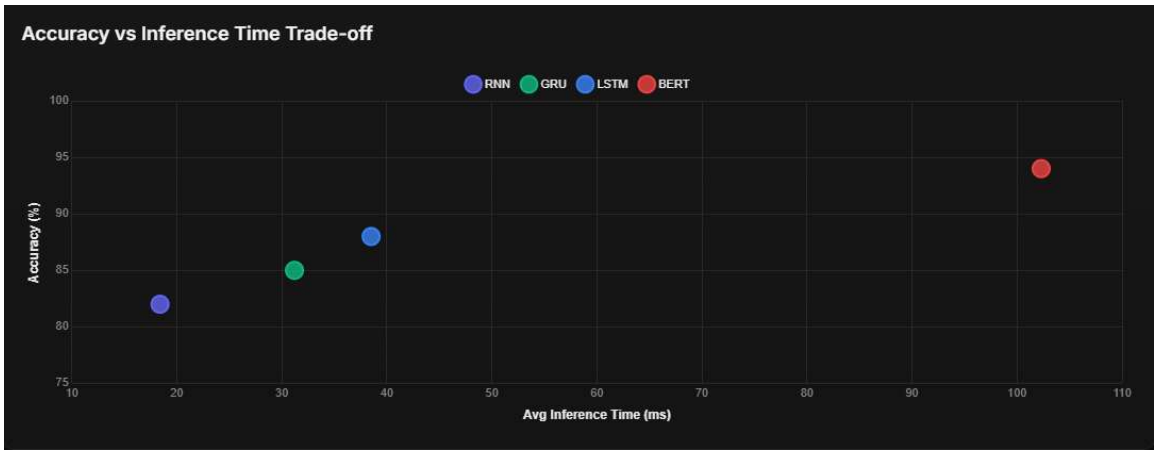
The average inference time for each model is presented in this graph, illustrating the time taken by a model to produce a prediction of sentiment once the input has been received. A lower inference time indicates a quicker performing model. Following the data, RNN has the lowest inference time (approximately 20 ms), and GRU and LSTM take slightly longer due to extra gating mechanisms. BERT has the highest inference time (approximately 100 ms) due to its transformer architecture. In sum, although BERT made the most accurate and confident predictions, the RNN-based models have a significantly lower inference time, suggesting they would be ideal for a real-time or limited resource application.

5.3 Performance Radar:



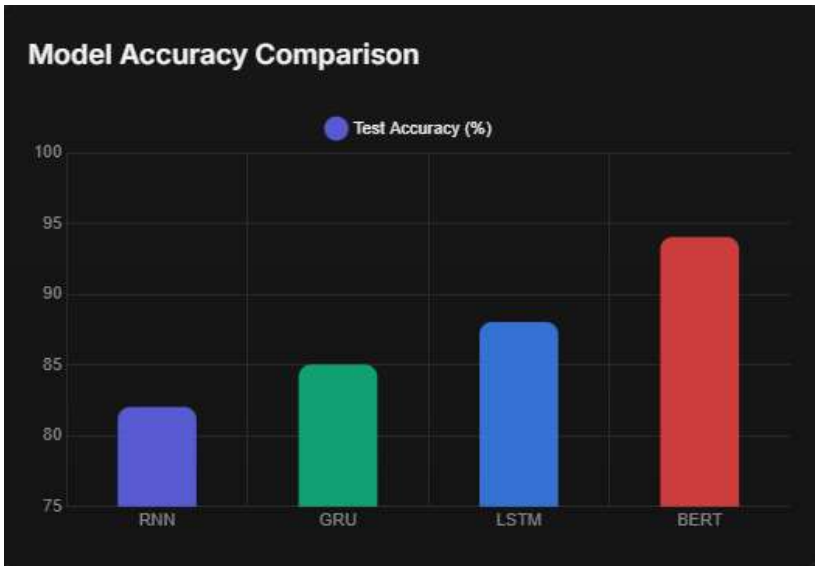
This radar chart examines the overall performance of the competing models in three dimensions: accuracy, speed, and compactness. The results suggest that BERT achieves the highest accuracy, but its larger model size contributes to slower speed and compactness. Conversely, RNN performs the best for speed and compactness, but with lower accuracy in comparison. GRU and LSTM models provide a fair compromise as they perform in a balanced fashion across all dimensions. Overall, the radar chart illustrates that BERT is the best predictor of quality, while RNN models remain more efficient for lightweight and faster predictions.

5.4 Accuracy vs Inference Time Trade-off:



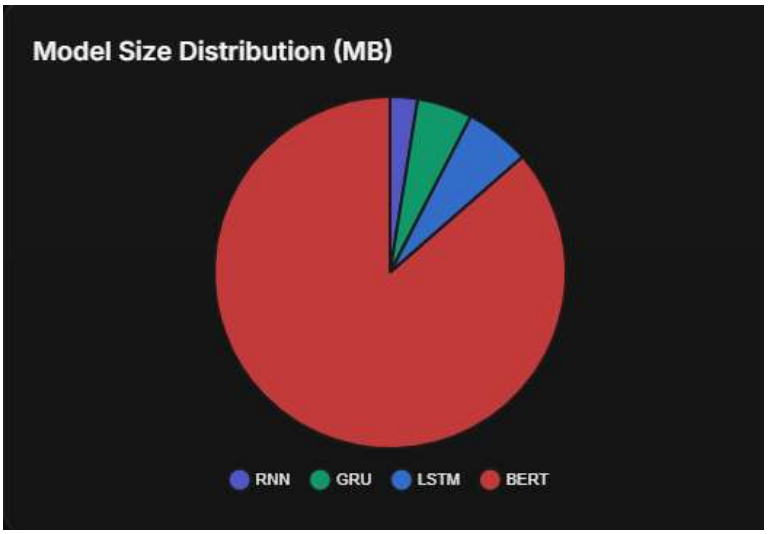
This graph illustrates the trade-off between model accuracy and inference time, showing how performance quality balances with processing speed. BERT achieves the highest accuracy (around 94%) but requires the longest inference time (around 100 ms), indicating its computational complexity. LSTM and GRU offer moderate accuracy (around 85–88%) with relatively lower inference times, showing an efficient balance. RNN provides the fastest inference (around 20 ms) but with the lowest accuracy. Overall, the plot highlights that higher model accuracy generally comes at the cost of slower response time.

5.5 Model Accuracy Comparison:



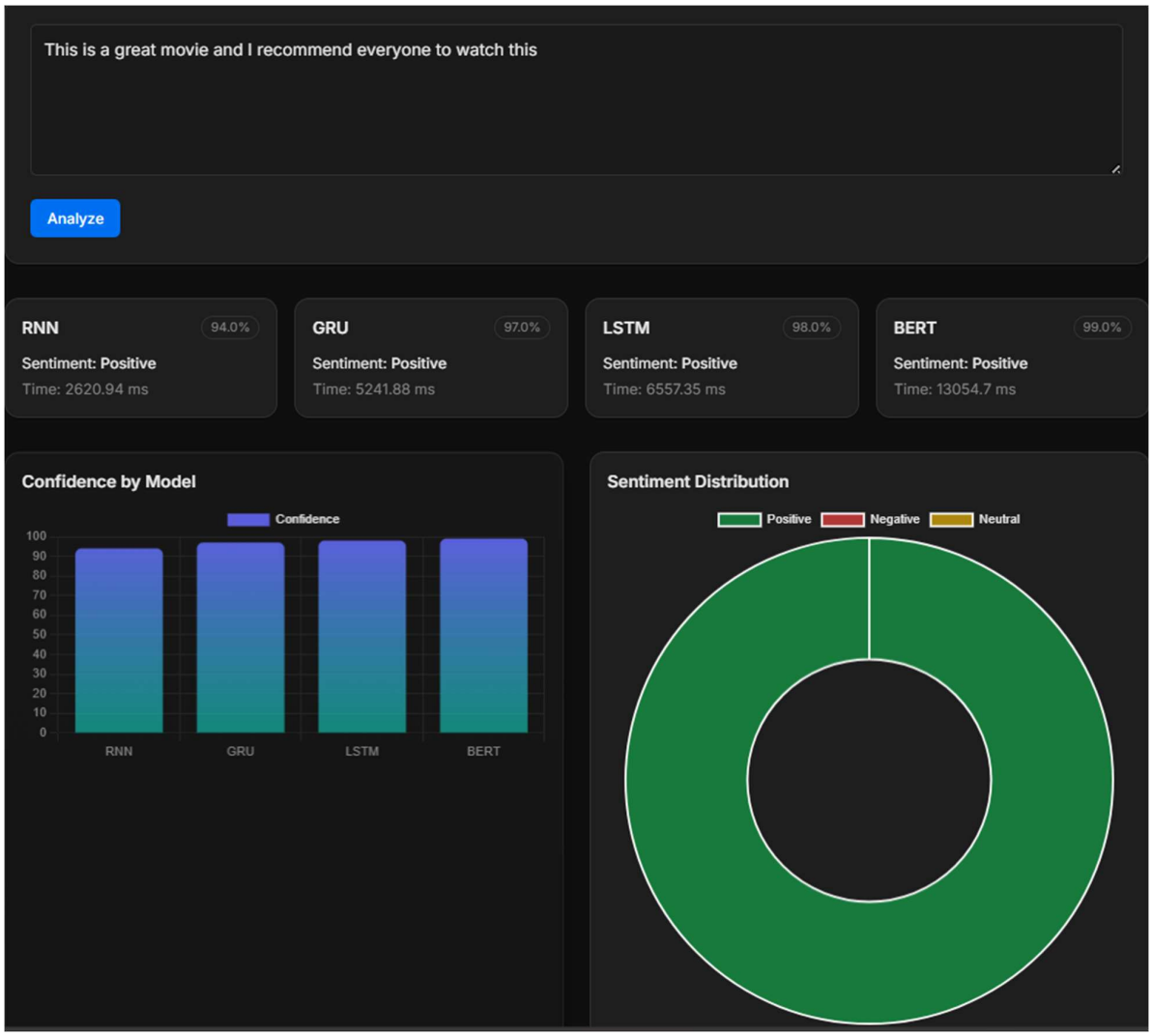
This graph compares the test accuracies of four deep learning architectures: RNN, GRU, LSTM, and BERT to highlight their performance differences. As seen, BERT achieves the highest accuracy (around 94%), reflecting its strong contextual understanding and advanced transformer-based design. LSTM and GRU show moderate accuracies (around 85–88%), indicating their efficiency in learning sequential patterns, while RNN records the lowest accuracy (around 82%), showing its limitations with long-term dependencies. Overall, the graph demonstrates that more advanced architectures like BERT provide a notable improvement in accuracy over traditional recurrent models.

5.6 Model Size Distribution:



This graph illustrates the model size distribution (in MB) for four deep learning architectures—RNN, GRU, LSTM, and BERT. It clearly shows that BERT occupies the largest portion of the total model size, indicating its significantly higher number of parameters and computational complexity. In contrast, RNN, GRU, and LSTM models contribute only small fractions, reflecting their relatively lightweight and simpler architectures. Overall, the chart highlights that while BERT delivers superior performance, it does so at the cost of substantially increased model size and resource requirements compared to traditional recurrent models.

5.6 Input based Analysis:



This visualization provides a comprehensive comparison of the four models: RNN, GRU, LSTM, and BERT, based on their sentiment prediction performance, confidence levels, and processing times. The top section shows that all models predicted a positive sentiment, with accuracy steadily improving from RNN (94%) to BERT (99%), reflecting the growing capability of more advanced architectures. The confidence by model bar chart indicates that all models exhibited high confidence, but BERT achieved the most consistent and reliable prediction confidence. The sentiment distribution chart on the right shows that the overall classification output across models is entirely positive, confirming strong model agreement on sentiment polarity. However, the inference time comparison reveals a clear trade-off between accuracy and computational cost—while RNN is the fastest (≈2620 ms), BERT is the slowest (≈13054 ms) due to its complex transformer-based structure. Overall, this analysis highlights that although all models perform well on positive sentiment detection, BERT provides the highest accuracy and confidence at the expense of longer inference time, whereas simpler models like RNN and GRU offer faster but slightly less precise results

6. Conclusion

The project successfully demonstrates the design and development of a Smart Feedback System that integrates sentiment analysis and content moderation using multiple deep learning models. By employing the IMDb dataset from Kaggle, the system analyzes textual feedback and compares the performance of RNN, GRU, LSTM, and BERT models on various evaluation metrics such as accuracy, confidence, inference time, and overall efficiency. The results clearly indicate that BERT, being a transformer-based model, provides the highest accuracy and confidence levels, making it the most reliable for nuanced sentiment understanding. However, this comes with increased computational cost and longer inference time, highlighting the trade-off between accuracy and speed. In contrast, RNN models offer faster predictions with minimal resource usage, making them more suitable for real-time applications, though at the expense of reduced accuracy. GRU and LSTM models balance performance and efficiency, serving as intermediate options that combine contextual understanding with manageable computational demands.

Overall, this project achieves its primary goal of building an intelligent, comparative sentiment analysis platform capable of evaluating multiple deep learning models within a unified interface. It provides valuable insights into model performance, scalability, and deployment efficiency, laying a strong foundation for future enhancements such as multi-language support, domain adaptation, or integration of advanced transformer architectures like RoBERTa or GPT-based classifiers.