Volume: 14 Issue: I Month of publication: January 2026
DOI: https://doi.org/10.22214/ijraset.2026.77036

Web App for Detecting AI Text Using NLP Techniques

Voda Akshitha[1], G. Praveen Babu[2]
[1]Post-Graduate Student (Data Sciences), Department of CSE, University College of Engineering, Science & Technology Hyderabad (UCESTH), Jawaharlal Nehru Technological University, Hyderabad, India
[2]Associate Professor, Department of CSE, UCESTH, Jawaharlal Nehru Technological University, Hyderabad, India

Abstract: The rapid growth of large language models has increased the presence of AI-generated text across social media, making it increasingly challenging to distinguish automated content from human contributions. This raises concerns regarding misinformation, artificial engagement, and the overall integrity of online discussions, creating a need for dependable detection mechanisms. In response, this project develops a web-based system that determines whether a given text sample originates from a human user or an AI model. The development process is organized into multiple stages, beginning with the cleaning and preprocessing of social media text using Natural Language Processing methods, including contextual filtering and sentiment-aware analysis. Classification is carried out using transformer-based models such as BERT, and additional hybrid configurations with LSTM or CNN layers are explored to capture deeper linguistic and sequential features. To support large-scale and multilingual usage, the system is backed by a scalable architecture capable of real-time processing. An interactive interface is provided to allow users to review predictions, observe analytical trends, and navigate the system efficiently. The inclusion of Explainable AI tools further clarifies model behaviour by highlighting the textual elements that influence each decision.

Keywords: Artificial Intelligence Text Detection, Natural Language Processing, BERT Embeddings, Hybrid Classification Model, Random Forest, Transformer Architecture, Explainable AI, SHAP, Multilingual Text Processing, Real-Time Web Application

I. INTRODUCTION

Advances in Artificial Intelligence have significantly transformed Natural Language Processing (NLP), resulting in language models that can generate text with remarkable fluency and contextual awareness. Modern Large Language Models (LLMs)—including GPT-4, Bard, Claude, and others are capable of producing responses, explanations, and written content that closely resemble human expression. As these systems become integrated into everyday digital interactions, distinguishing between machine-generated and human-authored text has become increasingly challenging.

This inability to easily verify the origin of digital content has implications in several areas. Educational institutions struggle to assess the originality of student work, online platforms encounter artificially generated posts that influence discussions, and news environments face the risk of fabricated narratives. Conventional plagiarism detection methods offer limited value in this setting because AI-generated text is produced from learned patterns rather than copied material, requiring deeper linguistic analysis for reliable detection.

To address this problem, the present work introduces a text classification system designed to determine whether a given sample is written by a human or generated by an AI model. The system employs transformer-based architectures, with BERT serving as the primary model for extracting contextual representations from text. Additional variations, such as combining BERT features with LSTM networks or Random Forest classifiers, are explored to improve classification consistency and adaptability. The framework supports multiple languages and processes text in real time through a scalable backend environment.

An interactive interface is developed to allow users to submit input text, receive classification results, and view supporting interpretability information. Explainable AI components, including SHAP visualizations, help clarify which portions of the input influenced the model’s decision, promoting transparent and informed use of the system. The platform brings together preprocessing, model inference, and interpretability within a unified workflow aimed at analysing and differentiating text origin in modern digital spaces.

II. LITERATURE SURVEY

The growing capability of modern Natural Language Processing (NLP) systems, particularly those based on large language models, has made it increasingly challenging to distinguish AI-generated text from human writing. As a result, researchers have explored multiple techniques to improve the reliability of automated text-origin detection. Existing studies examine rule-based approaches, traditional machine learning algorithms, deep learning models, and transformer-based architectures to understand the linguistic and semantic cues that characterize AI-generated content. Reviewing prior work provides essential insights into current methodologies, limitations, and research gaps, forming the basis for developing more adaptable and accurate detection systems.

A. Chamathka Maddugoda (2023):

Title: A Comprehensive Review: Detection Techniques for Human-Generated and AI-Generated Texts
Maddugoda presents a broad review of existing detection approaches designed to differentiate between AI-generated and human-authored text. The study categorizes methods into rule-based systems, machine learning models, and transformer-based techniques. Key linguistic indicators such as stylometry, sentence structure, and syntactic irregularities are highlighted as useful markers for detection. The review also notes that the emergence of advanced models like BERT and GPT has increased the complexity of identifying AI-written content, emphasizing the need for more robust detection frameworks.

B. Preslav Nakov et al. (2023):

Title: *AI-Generated vs. Human Text: Introducing a New Dataset for Benchmarking and Analysis*
Nakov and colleagues introduce a benchmark dataset designed for evaluating the performance of AI-text detection systems across multiple genres. Using transformer-based models such as RoBERTa and other GPT-style detectors, the study analyzes classification accuracy and identifies linguistic patterns contributing to model performance. The authors observe that while these models achieve strong results on many text categories, they struggle with highly coherent or refined AI-generated samples, highlighting the need for detectors with better generalization capabilities.

C. Samet Oymak et al. (2021):

Title: Detection of Malicious Bots on Twitter through BERT Embeddings
Oymak's work focuses on identifying malicious bot accounts on Twitter using contextual embeddings extracted from BERT. The study combines BERT embeddings with Random Forest classifiers to differentiate between bot-generated and human tweets. Findings suggest that the semantic information captured by BERT significantly improves classification accuracy, especially when paired with traditional machine learning methods. This approach demonstrates the effectiveness of combining deep learning representations with ensemble techniques for detecting suspicious online activity.

III. EXISTING WORK

Current approaches for detecting AI-generated text primarily rely on rule-based techniques or shallow machine learning models that analyze surface-level features such as word frequency, sentence structure, and stylistic patterns. Although some systems employ fine-tuned transformer models like BERT, their capabilities often remain limited in terms of real-time processing, multilingual handling, and interpretability. Many existing tools struggle when evaluating high-quality, human-like text produced by advanced language models, resulting in inconsistent or unreliable predictions in dynamic environments such as social media. Furthermore, these systems typically lack interactive dashboards, confidence scores, or monitoring functionality, reducing their usefulness for analysts and content moderators who require fast, transparent decision support.

A. Limitations of Existing System

1) Many models fail to accurately detect text generated by advanced language models such as GPT-4, which closely resembles human writing.
2) Most existing solutions are language-dependent and show reduced accuracy for multilingual or region-specific text.
3) Users often receive predictions without explanations, resulting in low interpretability and reduced transparency.
4) Real-time detection is rarely supported, making current systems unsuitable for environments requiring immediate moderation.
5) The absence of user-friendly interfaces, dashboards, and interactive monitoring features limits their practical usability, especially for non-technical users.

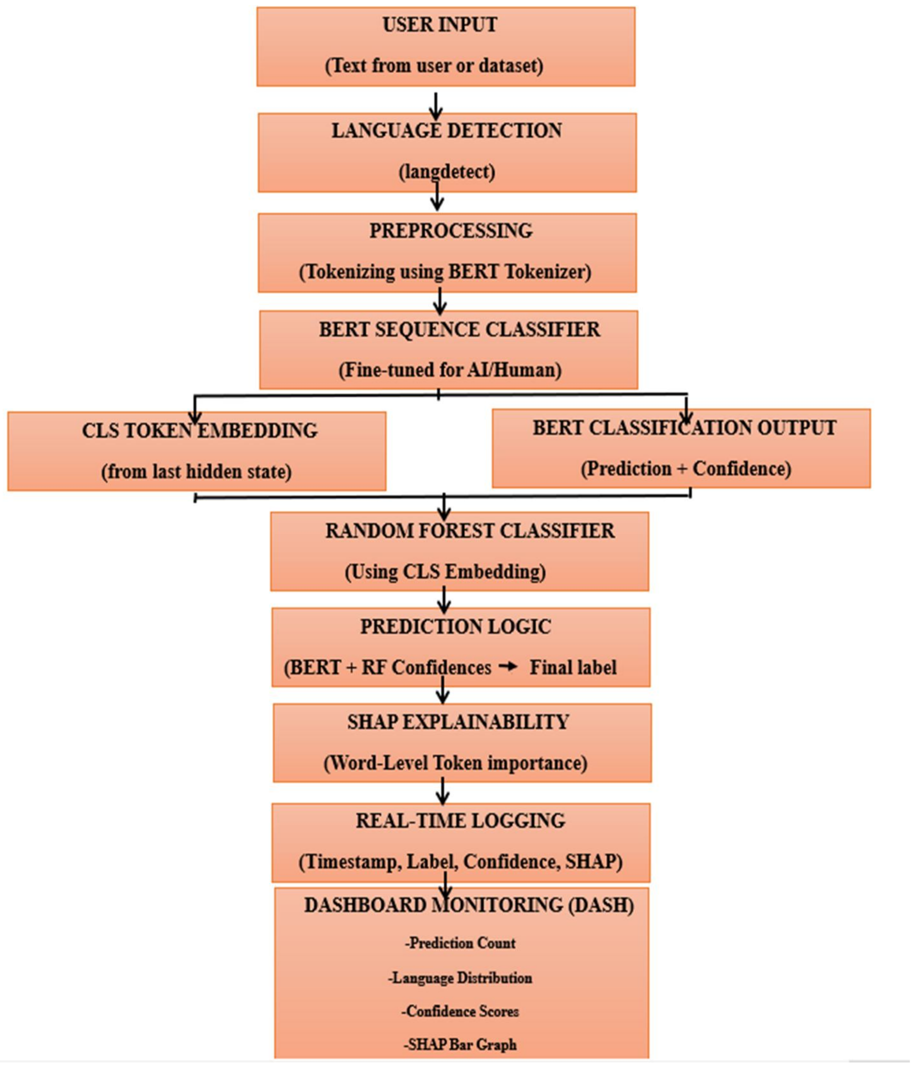
IV. PROPOSED WORK

The proposed system provides a real-time and scalable framework for distinguishing AI-generated text from human-written content. It uses BERT to extract contextual features and enhances classification through ensemble learning with a Random Forest model. A hybrid BERT–LSTM architecture is also examined to capture sequential patterns in text. The system supports multilingual inputs, includes SHAP-based explanations for interpretability, and is deployed through a Flask backend with a Dash dashboard for interactive monitoring. This design ensures accuracy, transparency, and practical usability for real-world applications.

A. Objectives

1) Utilizes transformer-based models like BERT to capture deep semantic and contextual information, improving detection accuracy.
2) Ensemble learning with Random Forest increases robustness and reduces the risk of overfitting.
3) Supports multilingual text, making it adaptable to diverse user environments and platforms.
4) Enables real-time predictions through a Flask backend, supporting fast and efficient processing.
5) Incorporates Explainable AI using SHAP to highlight influential words or tokens, enhancing transparency.
6) Provides a user-friendly, web-based dashboard for monitoring results, predictions, and confidence levels.
7) Scalable design allows the system to handle larger datasets and continuous input streams effectively.

V. SYSTEM ARCHITECTURE



VI. METHODOLOGY

The methodology outlines the complete workflow for developing the text classification system, including data collection, preprocessing, model training, evaluation, and deployment. Various NLP techniques, such as noise removal, text normalization, language detection, WordPiece tokenization, lemmatization, sequence preparation, and contextual embedding extraction, are applied to convert raw text into structured representations. Transformer-based models, hybrid neural architectures, and ensemble learning methods are employed to capture semantic information and stylistic patterns essential for reliable classification. The methodology ensures strong performance, multilingual compatibility, interpretability through SHAP, and smooth real-time interaction supported by a web-based interface.

A. Data Collection and Preprocessing

Text samples were gathered from various social media platforms, containing both human-written and AI-generated content. The preprocessing stage involved cleaning unwanted elements such as URLs, emojis, symbols, and excess spacing. Additional steps included lowercasing, tokenization, stopword removal, and automatic language identification. These processes ensured that the dataset remained consistent and suitable for model development.

B. Model Selection and Training

Multiple model architectures were examined, including a fine-tuned BERT classifier, a BERT–LSTM hybrid, and an ensemble that pairs BERT-derived [CLS] embeddings with a Random Forest model. The ensemble configuration was chosen for its improved robustness and ability to generalize across varied text patterns.

C. Model Evaluation

All models were assessed using widely accepted evaluation measures such as accuracy, precision, recall, and F1-score. This assessment enabled the selection of the most dependable model for deployment based on its consistent performance across validation data.

D. Explainability Component (SHAP)

To enhance transparency, SHAP was incorporated to highlight the influence of individual words on the model’s prediction. This interpretability layer helps users understand the reasoning behind classification outcomes.

E. Backend Development (Flask API)

A lightweight Flask API was developed to handle incoming text inputs and return classification results. The backend also manages language detection, ensemble decision-making, and the retrieval of system activity statistics through designated endpoints.

F. Real-Time Dashboard (Dash)

The system interface was created using Dash, providing visual summaries of prediction patterns, detected languages, confidence levels, and SHAP-based explanations. This dashboard enables real-time monitoring and smooth interaction for end users.

G. Feedback Module

A user feedback option is incorporated to confirm or dispute the system’s predictions. The feedback collected is stored and can later contribute to refining and improving the model’s performance.

H. Implementation

The system was implemented as a real-time text classification application using Python, with a Flask backend handling prediction request and a Dash dashboard visualizing results.
Jupyter Notebook was used for training all machine learning models with GPU support, after which the trained models were exported and integrated into the backend. The implementation ties together preprocessing, feature extraction, model prediction, explainability, and user interaction into a seamless workflow.
The core of the implementation relies on a multi-model algorithmic approach designed to improve accuracy, stability, and interpretability:

1) BERT (Bidirectional Encoder Representations from Transformers)

BERT is a transformer-based deep learning model designed to understand the context of a sentence by reading text in both forward and backward directions. It is highly effective for text classification tasks because it generates rich contextual embeddings that capture meaning, writing style, and semantic relationships. In this project, BERT was used as the primary classifier to distinguish AI-generated text from human-written text, utilizing the [CLS] token as a compact representation of the entire input.
We have applied BERT in our model as:
(i) Importing the BERT tokenizer and model from HuggingFace Transformers
(ii) Converting text into tokens and attention masks
(iii) Fine-tuning the model on labeled data

```
#BERT
from transformers import BertTokenizer, BertForSequenceClassification
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
bert_model = BertForSequenceClassification.from_pretrained('bert-base-uncased')
```

2) BERT + LSTM Hybrid Model

The BERT + LSTM architecture combines BERT’s contextual embeddings with the sequential learning capability of LSTM. BERT provides token-level embeddings, which are passed into an LSTM layer to capture long-range dependencies and writing flow. This model was tested to evaluate whether sequential patterns differ between AI and human writing. However, due to lower performance and inconsistency, it was not selected for the final deployment.
We have applied the BERT + LSTM model as:
(i) Extracting BERT embeddings for each token
(ii) Passing embeddings into an LSTM layer
(iii) Training the classifier on top of LSTM outputs

```
# BERT + LSTM Hybrid Model
bert_embeddings = bert_model(input_ids).last_hidden_state
lstm_output = LSTM (units=128)(bert_embeddings)
classifier.fit(lstm_output, Ytrain)
```

3) Random Forest Classifier (Using BERT Embeddings)

Random Forest is an ensemble algorithm that builds multiple decision trees and selects the final output based on majority voting. In this system, it was used not on raw text, but on the 768-dimensional BERT [CLS] embeddings. This allowed the model to combine BERT’s deep semantic understanding with Random Forest’s stability and interpretability. It helped reduce misclassifications, especially on borderline or ambiguous inputs.
We have applied Random Forest in our model as:
(i) Extracting BERT [CLS] embeddings
(ii) Creating a RandomForestClassifier object
(iii) Fitting it on the embedding vectors

```
from sklearn.ensemble import RandomForestClassifier
RF = RandomForestClassifier(n_estimators=100, random_state=0)
RF.fit(bert_cls_embeddings_train, Ytrain)
```

4) BERT + Random Forest Ensemble

The ensemble makes predictions by combining the outputs of both models, resulting in more reliable and consistent classification across diverse text inputs. This approach was chosen as the final system due to its strong generalization and reduced error rates.
We have applied the Ensemble approach as:
(i) Getting prediction probabilities from BERT

(ii) Getting prediction probabilities from Random Forest
(iii) Combining the results via weighted voting

```
bert_pred = bert_model.predict(Xtest)
rf_pred = RF.predict(bert_cls_embeddings_test)
# Ensemble logic
final_pred = (bert_pred + rf_pred) / 2
```

VII. PERFORMANCE EVALUATION

The system was evaluated using three different model configurations: a fine-tuned BERT classifier, a hybrid BERT + LSTM model, and a Random Forest classifier trained on BERT [CLS] embeddings. Their performance was assessed using accuracy, precision, recall, F1-score, and confidence stability across several test samples.

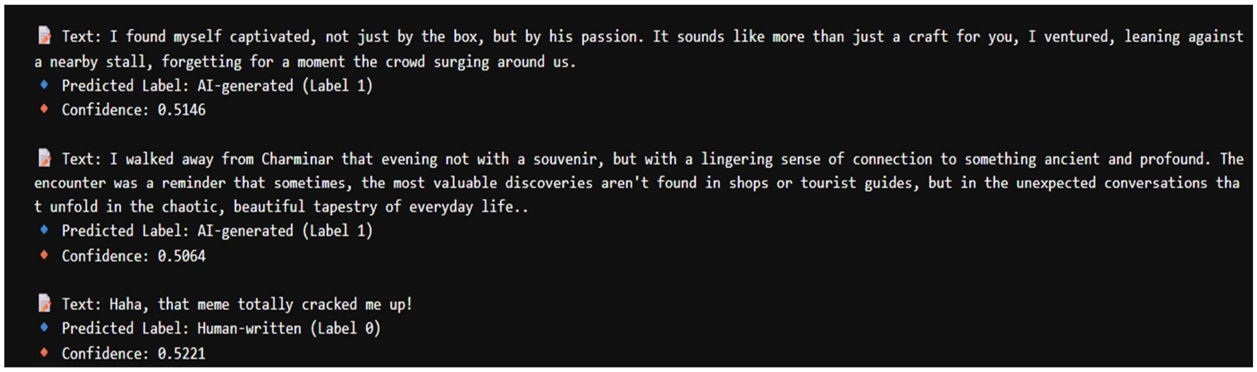
A. Model Performance Comparison

1) BERT (Fine-Tuned Sequence Classifier)

BERT demonstrated strong performance with consistently high confidence scores (>99% in most cases). It reliably distinguished AI-generated text from human-written text and produced stable predictions across diverse input samples.
Reason for Use: Deep contextual understanding and high accuracy.

2) BERT + LSTM Hybrid Model

Although the hybrid model was expected to capture long-range text patterns, it showed lower confidence and less stability compared to the fine-tuned BERT model. The confidence scores were often near 0.50, indicating uncertainty and reduced reliability.



```
📄 Text: I found myself captivated, not just by the box, but by his passion. It sounds like more than just a craft for you, I ventured, leaning against
a nearby stall, forgetting for a moment the crowd surging around us.
🔹 Predicted Label: AI-generated (Label 1)
🔸 Confidence: 0.5146

📄 Text: I walked away from Charminar that evening not with a souvenir, but with a lingering sense of connection to something ancient and profound. The
encounter was a reminder that sometimes, the most valuable discoveries aren't found in shops or tourist guides, but in the unexpected conversations tha
t unfold in the chaotic, beautiful tapestry of everyday life..
🔹 Predicted Label: AI-generated (Label 1)
🔸 Confidence: 0.5064

📄 Text: Haha, that meme totally cracked me up!
🔹 Predicted Label: Human-written (Label 0)
🔸 Confidence: 0.5221
```

Reason for Rejection: Lower confidence, less consistent predictions, did not outperform BERT alone

Fig 1: Confidence Score of BERT + LSTM Hybrid Model

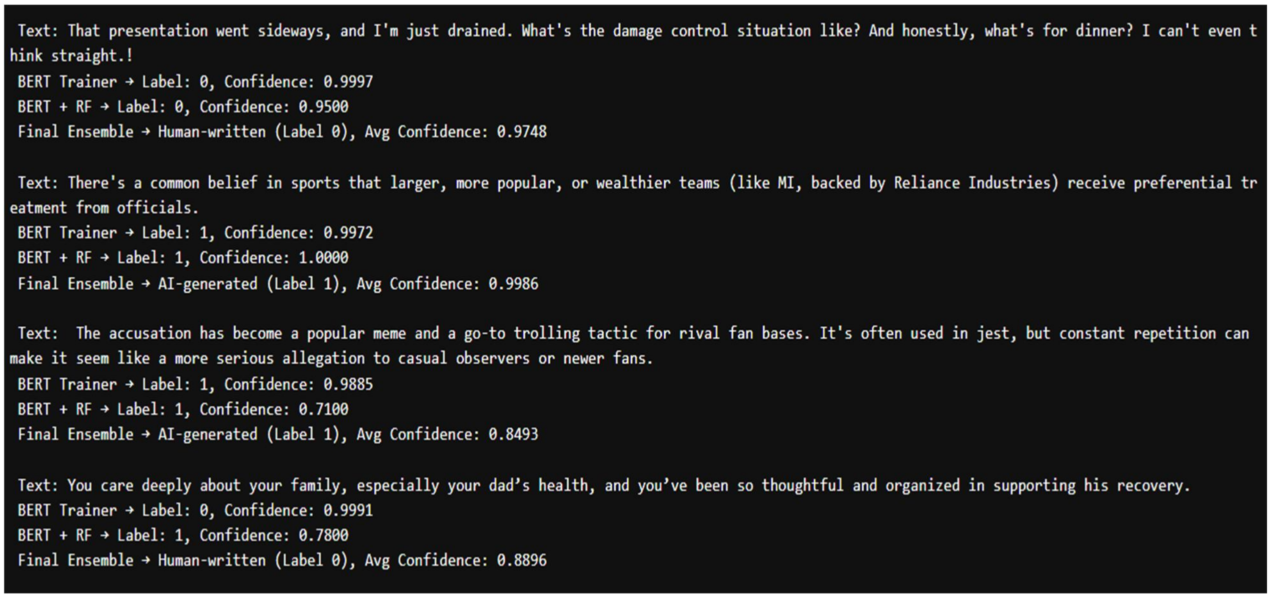
3) Random Forest (Using BERT [CLS] Embeddings)

Random Forest improved classification consistency by learning high-level semantic features from BERT embeddings. It produced sharp decision boundaries and stable results for ambiguous inputs.
Reason for Use: High interpretability and robustness when paired with BERT embeddings.

4) BERT + Random Forest (Ensemble Model)

The ensemble combined the strengths of both BERT and Random Forest, resulting in the most stable, reliable, and consistent classification output among all models.
The ensemble was therefore selected for deployment in the real-time web application.
Reason for use:

- More stable confidence than BERT alone
- Eliminated uncertainty seen in BERT+LSTM
- Improved robustness for borderline cases
- Highest overall reliability for real-time use



```
Text: That presentation went sideways, and I'm just drained. What's the damage control situation like? And honestly, what's for dinner? I can't even t
hink straight.!
BERT Trainer → Label: 0, Confidence: 0.9997
BERT + RF → Label: 0, Confidence: 0.9500
Final Ensemble → Human-written (Label 0), Avg Confidence: 0.9748

Text: There's a common belief in sports that larger, more popular, or wealthier teams (like MI, backed by Reliance Industries) receive preferential tr
eatment from officials.
BERT Trainer → Label: 1, Confidence: 0.9972
BERT + RF → Label: 1, Confidence: 1.0000
Final Ensemble → AI-generated (Label 1), Avg Confidence: 0.9986

Text:  The accusation has become a popular meme and a go-to trolling tactic for rival fan bases. It's often used in jest, but constant repetition can
make it seem like a more serious allegation to casual observers or newer fans.
BERT Trainer → Label: 1, Confidence: 0.9885
BERT + RF → Label: 1, Confidence: 0.7100
Final Ensemble → AI-generated (Label 1), Avg Confidence: 0.8493

Text: You care deeply about your family, especially your dad's health, and you've been so thoughtful and organized in supporting his recovery.
BERT Trainer → Label: 0, Confidence: 0.9991
BERT + RF → Label: 1, Confidence: 0.7800
Final Ensemble → Human-written (Label 0), Avg Confidence: 0.8896
```

Fig 2: Confidence Score of BERT + Random Forest (Ensemble Model)

VIII. RESULTS

To enable real-time evaluation and seamless user interaction, an interactive web-based dashboard was implemented using Dash. The dashboard presents model predictions, confidence scores from both the BERT classifier and the Random Forest model, the final ensemble decision, detected language, and SHAP-derived token importance. It also maintains a log of recent predictions, allowing transparent observation of system behavior over time. This interface demonstrates the deployment readiness of the system and provides clear analytical insight into classification trends through intuitive visual summaries.

The experimental findings indicate that the ensemble approach delivers superior stability, interpretability, and overall performance compared to the other evaluated architectures, making it the most suitable model for real-time text classification applications.

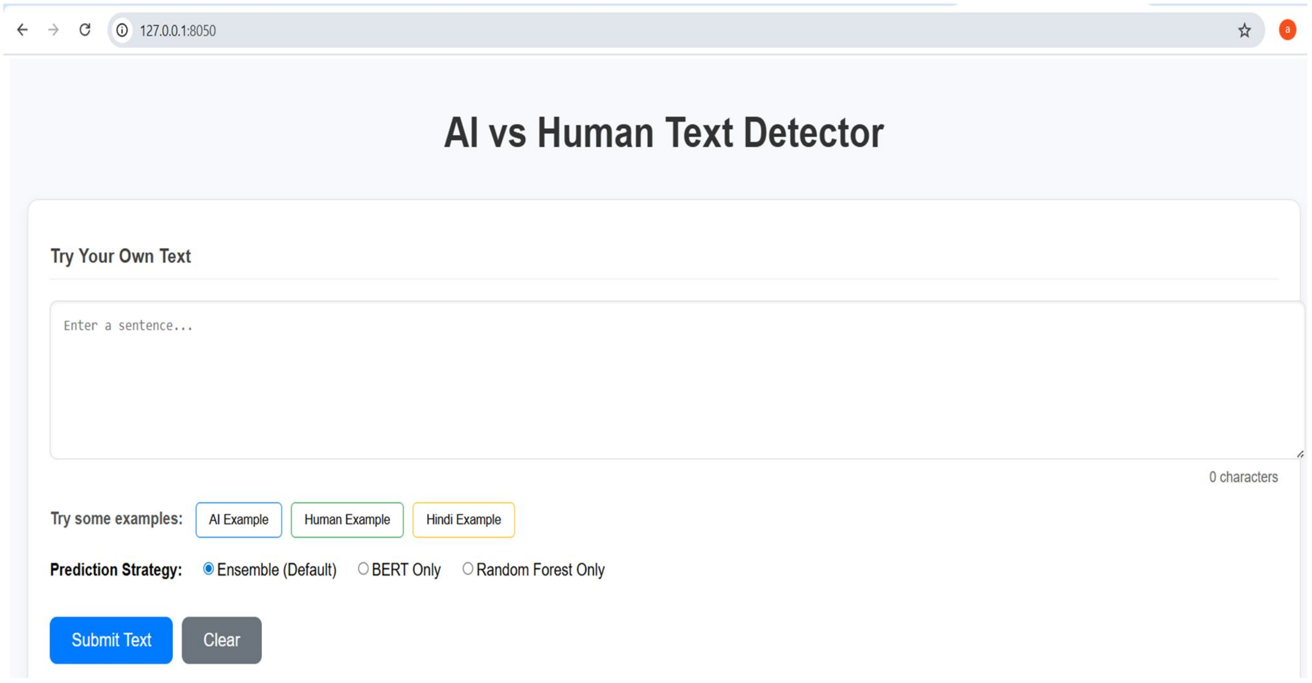
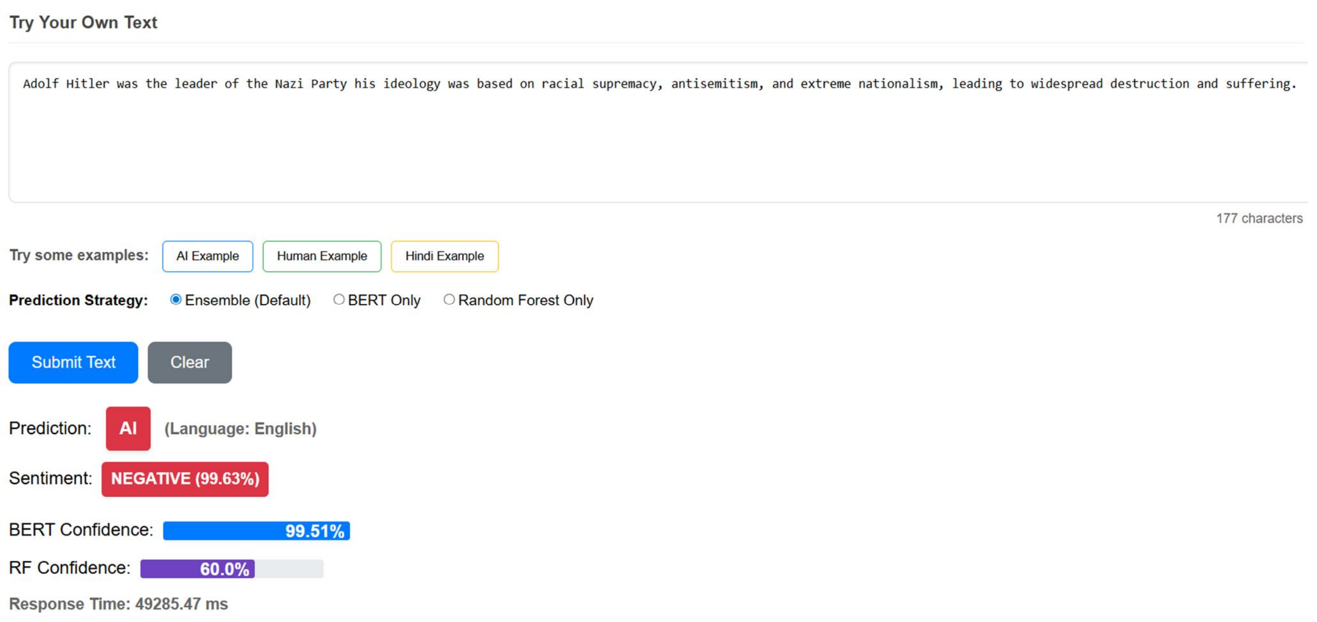


Fig 3: User Interface



Try Your Own Text

Adolf Hitler was the leader of the Nazi Party his ideology was based on racial supremacy, antisemitism, and extreme nationalism, leading to widespread destruction and suffering.

177 characters

Try some examples: AI Example | Human Example | Hindi Example

Prediction Strategy: [✓ Ensemble (Default)] BERT Only Random Forest Only

Submit Text | Clear

Prediction: AI (Language: English)

Sentiment: NEGATIVE (99.63%)

BERT Confidence: 99.51%

RF Confidence: 60.0%

Response Time: 49285.47 ms

Fig 4: User Prediction of AI generated Text

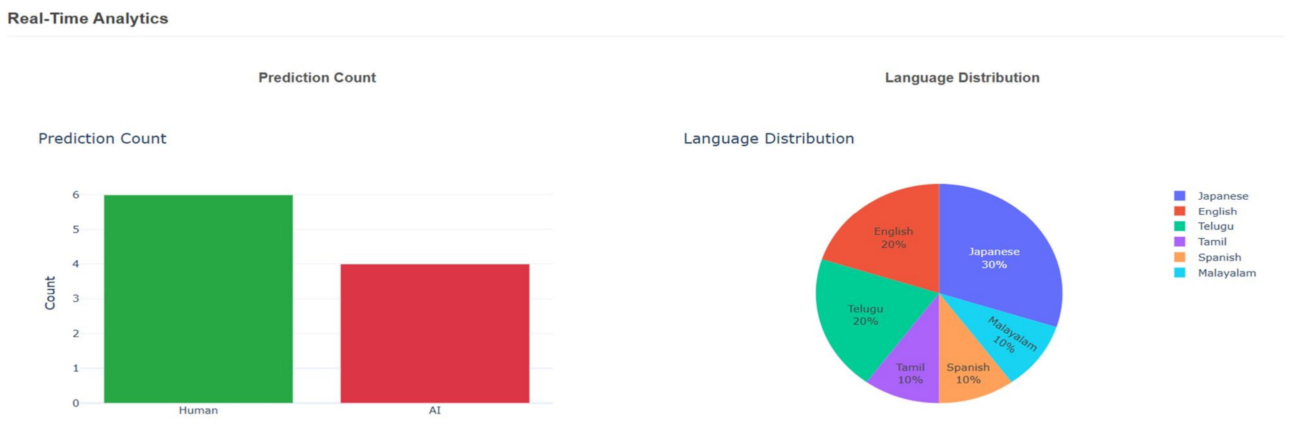


Fig 5: Prediction Count and Language Distribution

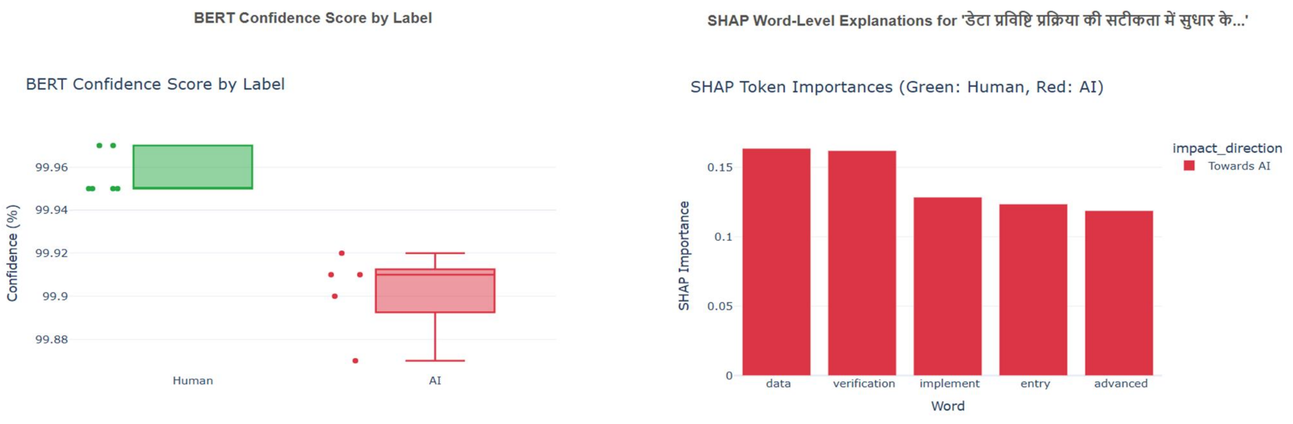


Fig 6: Confidence scores and SHAP word-level explanations

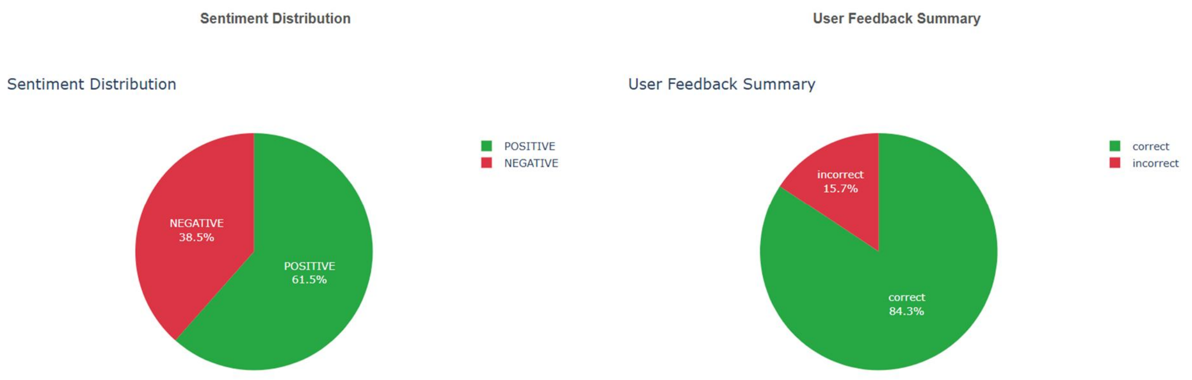


Fig 7: Sentiment distribution and User feedback summary

IX. CONCLUSION

This work presents a robust and real-time system for distinguishing between human-written and AI-generated text using a hybrid architecture that integrates a fine-tuned BERT model with a Random Forest classifier. The ensemble approach enhances prediction reliability and ensures consistent performance across diverse textual inputs. A key contribution of this work is the incorporation of SHAP-based explainability, which provides transparent, token-level insights into model decisions and strengthens user trust in the system's outputs.

The system is complemented by a real-time interactive dashboard developed using Dash, enabling clear visualization of predictions, confidence scores, and model behavior. A feedback mechanism allows users to validate or correct results, offering valuable support for potential future refinement. With a scalable Flask backend and an intuitive frontend, the framework demonstrates strong practical applicability in areas such as content moderation, academic integrity verification, and social media analytics.

A. Future Scope

Looking ahead, the system can be advanced to improve adaptability, multilingual robustness, and deployment scalability. As AI-generated text evolves, incorporating continuous learning and enhanced language modeling will be essential. Expanding accessibility and feature depth can further strengthen its practical value in real-world settings. The following developments may be considered for future enhancement:

1) Implementing online learning so the model can refine itself over time using stored user feedback.
2) Integrating region-specific transformer models to improve accuracy for non-English text.
3) Deploying the system on cloud platforms such as AWS or GCP for scalable, multi-user access.
4) Developing a browser extension for instant AI-text detection on social media and web pages.
5) Adding voice-to-text support to classify spoken content.
6) Incorporating additional modules such as bias detection, toxicity scoring, or sentiment analysis.
7) Extending compatibility across mobile and desktop platforms for broader accessibility.

REFERENCES

[1] Maddugoda, Chamathka. (2023). A Comprehensive Review: Detection Techniques for Human-Generated and AI-Generated Texts.
[2] A. Al Bataineh, R. Sickler, K. Kurcz and K. Pedersen, "AI-Generated Versus Human Text: Introducing a New Dataset for Benchmarking and Analysis," in IEEE Transactions on Artificial Intelligence, vol. 6, no. 8, pp. 2241-2252, Aug. 2025, doi: 10.1109/TAI.2025.3544183.
[3] Vyas, Piyush & Vyas, Gitika & Chennamaneni, Anitha. (2023). Detection of Malicious Bots on Twitter through BERT Embeddings-based Technique.
[4] Muhammad Irfaan Hossen Rujeedawa, Sameerchand Pudaruth and Vusumuzi Malele, "Unmasking AI-Generated Texts Using Linguistic and Stylistic Features" International Journal of Advanced Computer Science and Applications(ijacsa), 16(3), 2025. http://dx.doi.org/10.14569/IJACSA.2025.0160321.

00OO00