Journal of Advances in Information Technology, Vol. 16, No. 7, 2025

AI-Generated Text Detection and Source Identification

Anjana Priyatham Tatavarthi *, Faranak Abri *, and Nada Attar
Department of Computer Science, San José State University, San José, USA
Email: anjanapriyatham.tatavarthi@sjsu.edu (A.P.T.); faranak.abri@sjsu.edu (F.A.);
nada.attar@sjsu.edu (N.A.)
*Corresponding author

Abstract—**The use of advanced machine learning techniques to detect AI-generated text is a very practical application. The ability to distinguish human-written content from machine-generated text while identifying the source generative model helps address growing concerns about authenticity and accountability in digital communication. The differentiation of human-generated and AI-generated text is highly relevant to several applications, from news media to academic integrity, and is key to ensuring transparency and trust in content-driven environments. However, existing models are often insufficient to accurately detect AI-generated text and determine the specific AI source due to the complex nature of machine-generated content. To address this, it is essential to leverage state-of-the-art machine learning models and embedding techniques that can capture the subtle linguistic and contextual patterns of AI-generated text. In this study, experiments involving text classification were conducted to develop models capable of distinguishing AI-generated content from human-written text and identifying the specific AI model used, offering a multilayered approach to detection. The results demonstrate that the Long Short-Term Memory (LSTM) model with Bidirectional Encoder Representations from Transformers (BERT) embeddings outperformed other embedding techniques at the task of binary classification, achieving a score of 97% for both accuracy and F1 metrics. Additionally, this study illustrates the superior performance of pretrained transformer-based models compared to Recurrent Neural Network (RNN)-based models for four-class source identification, with Robustly optimized BERT approach (RoBERTa) achieving a score of 88% for both accuracy and F1 metrics. This highlights the advantage of leveraging powerful Large Language Models (LLMs) for the complex task of source identification, offering a more robust and scalable solution compared to traditional approaches.**

Keywords—**AI-generated text, Bidirectional Encoder Representations from Transformers (BERT) model, Bidirectional Long Short-Term Memory (BiLSTM), Deep Learning (DL), Large Language Models (LLMs), Long Short-Term Memory (LSTM), Machine Learning (ML), Robustly optimized BERT approach (RoBERTa) model, word embeddings**

I. Introduction

We live in an era where Artificial Intelligence (AI) writing tools can generate almost anything imaginable, from news articles and academic papers to poetry and novels. As researchers who are working at the intersection of AI and Natural Language Processing (NLP), we have witnessed generative capabilities evolve from crude text to sophisticated writing that even keen readers cannot distinguish from work by humans. This rapid advancement brings both excitement and concern. Although AI writing tools offer unprecedented possibilities for content creation, they also raise critical questions about authenticity and trust in our digital world [1].

Text classification has long been a central challenge in NLP, with AI models applied to a wide range of tasks from traditional problems such as phishing email detection [2] and fake review [3] identification to more complex issues like lie and deception detection [4]. These tasks not only test the limits of model accuracy but also underscore the importance of nuanced, context-aware understanding, which is highly relevant to the emerging challenges posed by AI-generated content.

Our study addresses a fundamental problem: how to reliably detect the difference between human-written and AI-generated text; and more importantly, determine which specific AI model generated the artificial text. Our research indicates that human readers, including highly qualified educators and content moderators, are unable to consistently classify human- and AI-authored content [5]. This challenge is further complicated by the rapid evolution of Large Language Models (LLMs), such as Generative Pre-trained Transformer (GPT) [6], Large Language Model Meta AI (LLaMA) [7], and Gemini [8] , which are capable of producing increasingly sophisticated and human-like content [9], [10]. As traditional detection methods struggle to keep pace with these advancements, our work aims to bridge this widening gap with more robust and precise identification techniques.

This problem becomes more complex when considering the nuances of different AI models. Each model has a fingerprint, that is, subtle patterns in how it constructs sentences, selects words, and maintains consistency [11]. These patterns are too subtle for human detection, but they may be identifiable with the proper computational approach. Therefore, our research focuses on developing frameworks that can not only separate human-written from AI-generated text, but also identify which particular AI model generated the content.

We employed machine learning techniques to address these challenges of AI-generated text detection. We utilized advanced neural network architectures such as Long Short-Term Memory (LSTM) and Bidirectional Long Short-Term Memory (BiLSTM), along with transformer models including Bidirectional Encoder Representations from Transformers (BERT) and Robustly optimized BERT approach (RoBERTa), to capture linguistic patterns that can be used to distinguish AI-generated text [12]–[15]. We also incorporated various processing techniques, including BERT and RoBERTa embeddings, to analyze textual patterns and

Manuscript received February 3, 2025; revised March 17, 2025; accepted May 13, 2025; published July 25, 2025.

characteristics to aid in the detection of AI-generated text and source model identification.

The main points of this study are as follows:

– Exploring embedding techniques (Word2Vec, one-hot, and pretrained embeddings) and analyzing their impact on model performance for binary AI-generated text detection
– Evaluating and comparing transformer models with traditional Recurrent Neural Networks (RNNs) for multiclass AI-generated text source identification
– Achieving high performance in four-class AI-generated text source identification using pretrained transformer-based models

This study is distinguished by its practical focus. While previous research treats AI text detection as solely a binary problem (human vs. AI), we tackle the more nuanced challenge of identifying specific AI sources. This is highly relevant, as different AI models may be used for varying purposes, some legitimate and others potentially malicious. Our system helps attribute text to its source, whether it is a human writer or a specific AI model, to assign context, authorship, and responsibility to each text.

This paper is structured as follows. Section II reviews existing research in the field of AI-generated text detection. Section III describes the datasets used in our experiments. Section IV outlines the step-by-step approach we followed to detect AI-generated text and identify the source model. Section V presents the experimental setup for the binary classification and multiclass classification models, along with their performance, and our analysis. Section VI includes a discussion of the results of our experiments, with insight on the effectiveness of different approaches and reasoning behind their performances. Finally, Section VII concludes the paper and suggests future work on this topic.

II. Related Work

This section reviews related studies on both binary and multiclass classification tasks regarding AI-generated text. A summary of the papers that involve binary classification is provided in Tables I, while Table II presents those that focus on multiclass classification.

A. Binary Classification

1) Classic machine learning models: Shijaku *et al.* [16] developed a model to detect ChatGPT-generated text using a dataset of human-written and ChatGPT-generated essays. They extracted various lexical features from the text, involving characters, words, sentences, digits, and part-of-speech tags. Feature selection was conducted using both the Term Frequency-Inverse Document Frequency (TF-IDF) technique and a set of handcrafted features to identify the most informative attributes. These features were then used to train an XGBoost classifier, achieving an accuracy of 0.96 with custom features and 0.98 with TF-IDF.

Luo *et al.* [17] developed a framework for AI-Generated Review Detection using Cumulative Probability (AGRDCP) for e-commerce reviews. The dataset they used contains human-written reviews from Yelp.com and AI-generated reviews generated by GPT-3 [18]. The AGRDCP framework leverages traditional and novel linguistic features such as perplexity and burstiness, along with two existing models, "roberta-large-openai-detector" and "chatgpt-detector-roberta," to produce a probability score indicating the likelihood that a review is AI-generated. This approach utilizes cumulative probability density and machine learning classifiers including k-Nearest Neighbors (KNN), Logistic Regression (LR), Random Forest (RF), Support Vector Machine (SVM), and AdaBoost for analysis. The results show that the proposed method significantly outperforms the existing baseline methods in detecting AI-generated reviews, with an F1-score of roughly 0.85.

Corizzo *et al.* [19] discussed using one-class machine learning models and linguistic features to detect AI-generated essays, focusing on the English and Spanish languages. The authors experimented with the following models: one-class SVMs, local outlier factor, and isolation forest, which were trained to differentiate between human- and AI-written content. The chosen dataset includes essays written by English and Spanish learners and their AI-generated counterparts. Their results demonstrated the effectiveness of this approach in identifying essays generated by AI, showing the strong potential of linguistic analysis and one-class models for this purpose, even without AI-generated examples during training.

Bao *et al.* [20] applied an efficient zero-shot method for detecting machine-generated text using conditional probability curvature. This probability is a new metric that can highlight word choice discrepancies between LLMs and humans. It was tested on diverse datasets and models and showed superior performance.

Deng *et al.* [21] proposed a detection framework for LLM-generated texts through a Gaussian Process (GP) Bayesian surrogate model. This approach enhanced their previous DetectGPT method, as it reduced the number of queries required for detection and thus improved efficiency without compromising accuracy. The experiments were performed on several datasets including XSum, SQuAD, and WritingPrompts, and using the GPT-2 [22] and LLaMA-65B [23] language models. Their results demonstrated the efficiency of this new approach, which required fewer queries yet achieved a better performance than DetectGPT.

2) Advanced machine learning models: Bhattacharjee [24] proposed a new direction in the detection of AI-generated news as an unsupervised task. Their idea relies on the Contrastive Domain Adaptation (ConDA) framework, which combines the power of standard domain adaptation with contrastive learning. Domain adaptation is a process of fine-tuning a model, which was previously trained on a source dataset, to be effective when applied to a target, usually unlabeled, dataset. Contrastive learning is a method that learns the characteristics that make two samples similar or dissimilar by directly comparing them. The authors aimed to apply this approach to identify domain invariant features, which can be effective for performing unsupervised detection. They used a dataset containing human-written news articles and AI-generated ones from various large language models. The results showed that ConDA outperformed all the baseline methods, with an improved 31.7% enhancement over the highest-performing existing approaches.

Gaggar *et al.* [25] evaluated models such as SVM, RoBERTa-base, and RoBERTa-large on the task of AI-

TABLE I: Similar Works in Binary Classification of AI-Generated Text

Paper	Dataset	Features	Model	Metrics	Results
[16]	ChatGPT (TOEFL essays)	Lexical Features	XGBoost	Accuracy	0.96
[17]	Online Review Dataset	LIWC, Readability, Linguistic features	AdaBoost	F1 Score	0.83
[19]	ChatGPT Essays	Textual, Repetitiveness, Semantic, POS	OneClassSVM	F1 Score	0.7283
[20]	XSum, SQuAD, WritingPrompts, WMT16	Conditional Probability Curvature	FAST-GPT	Accuracy	0.78
[21]	XSum, Wikipedia paragraphs	Conditional Probabilities of Tokens	Zero-shot Conditional Probabilities	AUC-ROC	0.929
[24]	TuringBench4 (news articles)	RoBERTa Embeddings	RoBERTa + MLP	F1 Score	0.92
[25]	Twitter, PubMedQA, SQuAD	Byte-level BPE Tokenizer	RoBERTa-large	AUC-ROC	0.9576
[26]	ChatGPT on various topics	Perplexity, Semantic, Document, Readability	Random Forest	Accuracy, F1 Score	0.95, 0.95
[27]	ALTA Shared Task 2023	Various Pretrained Embeddings	Ensemble(ALBERT, ELECTRA, RoBERTa, XLNet)	Accuracy	0.9694
[28]	M4 (multi-lingual corpus)	GLTR, BERT Embeddings, Semantic features	XLM-R Classifier	F1 Score	0.82
[29]	OpenLLMText (GPT3.5, PaLM, LLaMA, GPT2)	Next Token Probabilities	T5	Accuracy	0.96
[30]	TweepFake (real Twitter posts)	TF-IDF, BERT, Char Embeddings	RoBERTa	Accuracy	0.896
[31]	Wikipedia (SQuAD), XSum, HC3	Proxy Perplexities from N-gram Token Probabilities	t LLMDet	Precision	0.98

TABLE II: Similar Works in Multiclass Classification of AI-Generated Text

Paper	Dataset	Features	Model	Metrics	Target Classes
[32]	DIALOG-22 RuATD	Textual Features, Embeddings	Ensemble(Various authors)	Accuracy = 0.829	4
[33]	Essays, Poems, Stories, Code	CountVectorizer, Tfidf	Random Forest	Accuracy = 0.9574, F1 = 0.83	3

generated text detection. They used various datasets including Twitter Sentiment, Football Commentary, Project Gutenberg, PubMedQA, and SQuAD. The study found that detection effectiveness is significantly influenced by sentence length.

Chen *et al.* [29] proposed a method to detect text generated by LLMs by reframing the classification task as a next-token prediction problem instead. Instead of adding a separate classification layer, they simply fine-tuned a base language model, specifically the Text-to-Text Transfer Transformer (T5), to perform this task. Their dataset, OpenLLMText, comprises approximately 340,000 text samples from humans and various LLMs, including GPT-3.5 [18], Pathways Language Model (PaLM) [34], LLaMA [7], and GPT-2 [22]. The proposed framework, T5-Sentinel, outperformed an alternative variant that uses an attached classifier (T5-Hidden) in terms of weighted F1 score. The authors found that T5-Sentinel effectively distinguishes between texts from different LLMs and human-written texts, showing improvements in AUC, accuracy, and F1 scores over baseline detectors.

Fagni *et al.* [30] presented a comprehensive approach for identifying machine-generated text on social networks, namely Twitter. The authors formed the TweepFake dataset by collecting deepfake tweets posted on Twitter by 23 bots imitating 17 human accounts using methods such as Markov chains, RNN, LSTM, and GPT-2. The dataset contains 25,572 tweets in all, and contains equal parts human- and bot-generated samples. The study evaluates 13 deepfake text detection methods, which leverage techniques including basic machine learning classifiers like bag of words with TF-IDF, deep learning models, and fine-tuning transformer-based classifiers. Among all of the methods, the RNN-based detector was the best-performing model.

Wu *et al.* [31] presented LLMDet, a tool for detecting text generated by LLMs such as GPT-2, OPT, and LLaMA. LLMDet utilizes the next-token probabilities of n-grams to calculate a proxy perplexity for each LLM. This helps to identify the source model of a given text. The authors used various datasets such as Wikipedia, SQuAD, and XSum, with 32,000 samples per model. LLMDet achieved a precision of 0.98 and demonstrated high performance in differentiating human- and AI-generated texts, thereby outperforming existing methods.

B. Multiclass Classification

Maloyan *et al.* [32] focused on the binary task of identifying AI-generated text, and also the multiclass classification task of determining the origin of the generated text in the context of the DIALOG-22 RuATD challenge. They used an ensemble method of different pretrained models such as RoBERTa, Decoding-enhanced BERT with disentangled attention (DeBERTa), and RuBERT, along with attention mechanisms, and achieved an accuracy of 0.62 for multi-class classification.

Hayawi *et al.* [33] presented a comprehensive study on techniques for distinguishing human-written and AI-generated texts by GPT and BARD across multiple genres, including essays, stories, poetry, and Python code. The study analyzed texts for classification based on the source by utilizing Machine Learning (ML) such as Random Forest, Support Vector Machine, and Logistic Regression, and Deep Learning (DL) models such as Long Short-Term Memory networks. Results indicated varying levels of accuracy across models, with SVM performing best, with an F1 score of 0.74 on the essays dataset, 0.81 on the stories dataset, and 0.89 on the poetry dataset.

III. Dataset Description

This section will describe the two datasets used in this study: one for the task of binary classification and one for the task of multiclass classification.

A. Binary Classification Dataset

The **GPT-Wiki-Intro Dataset** [35] was specifically curated to benchmark the performance of models in differentiating between human-authored and machine-generated text. This dataset involves the domain of Wikipedia introductions, which implies that the textual topics are extensive and varied.

Composition: The GPT-Wiki-Intro dataset contains two types of text: authentic Wikipedia introductions and synthetic introductions generated by the GPT3 Curie model. This dataset consists of 150,000 samples spanning a diverse range of subjects. Each sample is represented by both a human-written Wikipedia introduction and a corresponding AI-generated counterpart.

Generation Methodology: A specific prompt structure was used for the generation of artificial samples for the dataset. Each prompt begins with a directive to produce a 200-word Wikipedia-style introduction on a given topic, followed by the first seven words of the actual Wikipedia introduction for that topic. This prompt structure ensures consistency in the starting point of the generated content. It also provides a controlled variable for comparing human and AI-generated text.

Data Schema: The GPT-Wiki-Intro dataset was structured for ease of use and analysis. Important columns include Wikipedia URLs, titles, both the Wikipedia and generated introductions, and various length metrics, such as the number of words in the title and introduction. This detailed schema allowed for granular analysis and comparison between human-written and AI-generated texts. Fig. 1 illustrates the word count distributions for human-written and AI-generated texts within the dataset.

B. Multiclass Classification Dataset

The **OpenLLMText Dataset** [29] was designed for multiclass classification, consisting of approximately 300,000 text entries distributed across four target classes: *Human, GPT2-XL, LLaMA, and PaLM*. This dataset integrates binary classification datasets from [29] into a unified multiclass dataset for the identification of AI-generated text sources. The dataset was stored in *JSONL (JSON Lines)* format. Table III provides an overview of the dataset's composition.

TABLE III: OpenLLMText Dataset Composition

Source	**Target Class**	**Entries**	**Description**
OpenWebText	Human	68,984	Human-written text randomly selected from user-generated content on Reddit prior to 2019.
PaLM	PaLM	57,410	Paragraph-by-paragraph rephrasing of human-written data by Pathway Language Model (PaLM).
LLaMA-7B	LLaMA	65,991	Paragraph-by-paragraph rephrasing of human-written data by LLaMA-7B.
GPT2-XL	GPT2	82,932	Data adapted from the GPT-2 output dataset released by OpenAI (GPT2-XL).

Fig. 2 illustrates the distribution of classes in the multiclass classification dataset.

Fig. 3 showcases the word count distributions for texts generated by the different models (GPT2, PaLM, and LLaMA) and the human-generated texts. Each histogram represents the density of word counts, providing insight into the variation in lengths of the generated and written texts. For instance, the human-generated texts appear to have a relatively uniform distribution while the generative models, especially LLaMA and PaLM, exhibit notable peaks.

IV. Methodology

This section outlines the step-by-step approach followed in this study to detect AI-generated text and identify the source model. We divided the methodology into the following stages: data collection, embedding techniques, model selection and implementation, training and hyperparameter tuning, and evaluation metrics.

A. Data Collection

Datasets: Two datasets were used in this study: GPT-Wiki-Intro and OpenLLMText.

- *GPT-Wiki-Intro*: Used for binary classification, contains human-written and GPT3 Curie-generated Wikipedia-style introductions.
- *OpenLLMText*: Used for multiclass classification, encompasses texts generated by GPT2, LLaMA, PaLM, and human authors.

B. Embedding Techniques

We employed three embedding methods to represent the textual data: one-hot, Word2Vec, and BERT.

- **One-Hot Embeddings:** Simple binary representations used as a baseline.
- **Word2Vec Embeddings:** Dense vector representations that capture semantic relationships between words [36].
- **BERT Embeddings:** Contextual embeddings generated using the pretrained Bidirectional Encoder Representations from Transformers (BERT) [37] model, specifically `bert-base-uncased` from Hugging Face.

C. Model Selection and Implementation

We utilized the following frameworks to perform the task of classification: RNNs and transformer-based models.

1) Recurrent Neural Networks (RNNs): We utilized LSTM and BiLSTM for their ability to capture sequential dependencies in text.

LSTM networks incorporate cell-level gating mechanisms to control the flow of information through time steps, helping them capture long-term dependencies in sequences. Eq. (1) describe how the LSTM selectively forgets, updates, and outputs information at each time step using these gating mechanisms.

LSTM Cell Equations:

$$\begin{aligned}
f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\
i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\
\tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \\
C_t &= f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \\
o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\
h_t &= o_t \cdot \tanh(C_t)
\end{aligned} \tag{1}$$

The Bidirectional LSTM (BiLSTM) extends the LSTM architecture by processing the input sequence in both forward and backward directions. This allows the model to capture context from both past and future tokens, which is especially useful for tasks like text classification and named entity recognition.

2) Transformer-based models: We chose to fine-tune the BERT [37] and RoBERTa [38] in order to leverage their pretrained contextual embeddings and self-attention mechanisms for this domain-specific task. Transformers rely on self-attention mechanisms to compute contextualized token representations by attending to all positions in the input sequence. Eqs. (2) and (3) describe scaled dot-product and multi-head attention, which are instrumental in these models.

Scaled Dot-Product Attention:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2}$$

where:

- Q, K, V are the query, key, and value matrices derived from the input embeddings.
- d_k is the dimensionality of the keys (used for scaling).

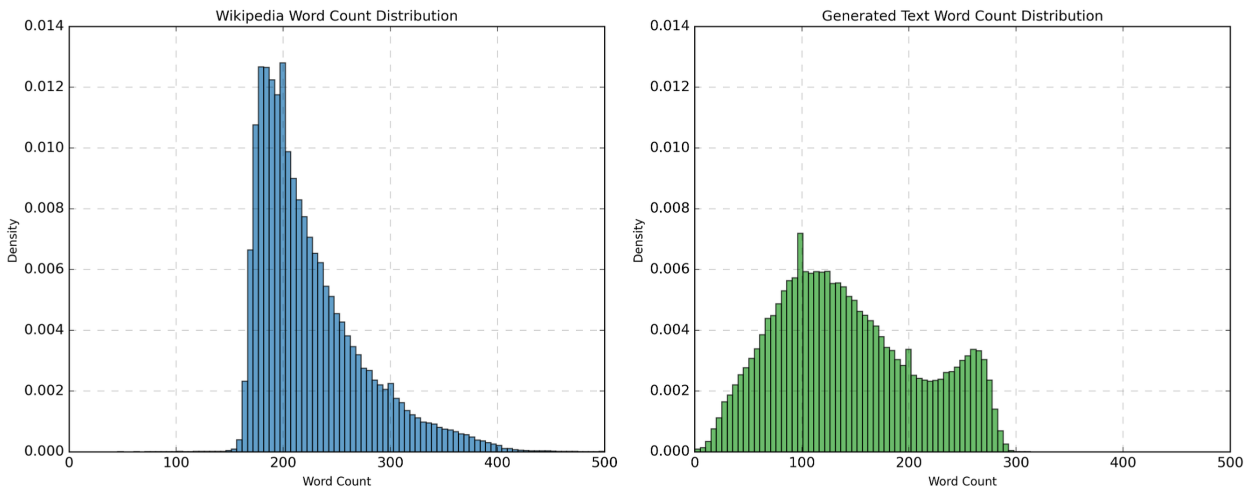

Fig. 1. Distribution of word counts in human-written and AI-generated Texts in the binary GPT-Wiki-Intro dataset.

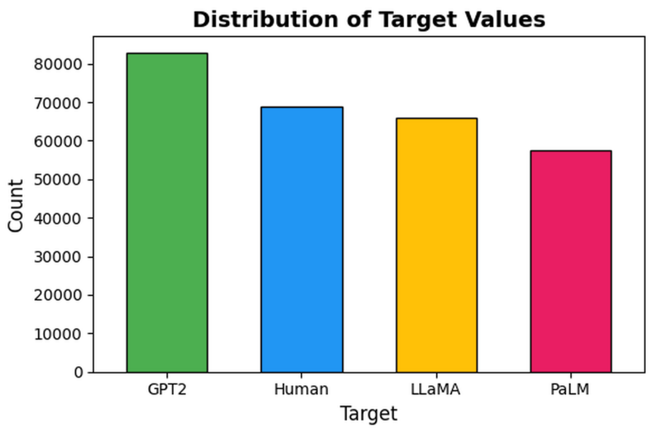

Fig. 2. Distribution of target values in the multiclass OpenLLMText dataset.

Multi-Head Attention:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O \quad (3)$$

$$\text{Where: head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

BERT, which stands for Bidirectional Encoder Representations from Transformers, is trained using Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). In contrast, RoBERTa, which stands for A Robustly Optimized BERT Pretraining Approach, improved upon BERT by removing NSP and using longer sequences for training, larger batch sizes, and a larger dataset. RoBERTa also dynamically changes the masking pattern during training, making it better suited to capture richer contextual information.

D. Training and Hyperparameter Tuning

Training Process: We split the datasets into training, validation, and test sets to support the training of robust models.

Hyperparameter Tuning: Model hyperparameters such as batch size, learning rate, number of layers, and dropout rate were optimized using grid search to further enhance model performance.

E. Evaluation Metrics

We evaluated the models based on the following standard classification metrics: accuracy, precision, recall, and F1-score. Note that in the following metric equations, TP, TN, FP, and FN stand for True Positives, True Negatives, False Positives, and False Negatives, respectively.

- **Accuracy:** Measures the proportion of correctly classified samples (Eq. (4)).

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (4)$$

- **Precision:** Evaluates the correctness of positive predictions (Eq. (5)).

$$\text{Precision} = \frac{TP}{TP + FP} \quad (5)$$

- **Recall:** Measures the ability to identify all relevant (positive) samples (Eq. (6)).

$$\text{Recall} = \frac{TP}{TP + FN} \quad (6)$$

- **F1-Score:** Represents the harmonic mean of precision and recall to balance their trade-offs (Eq. (7)).

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (7)$$

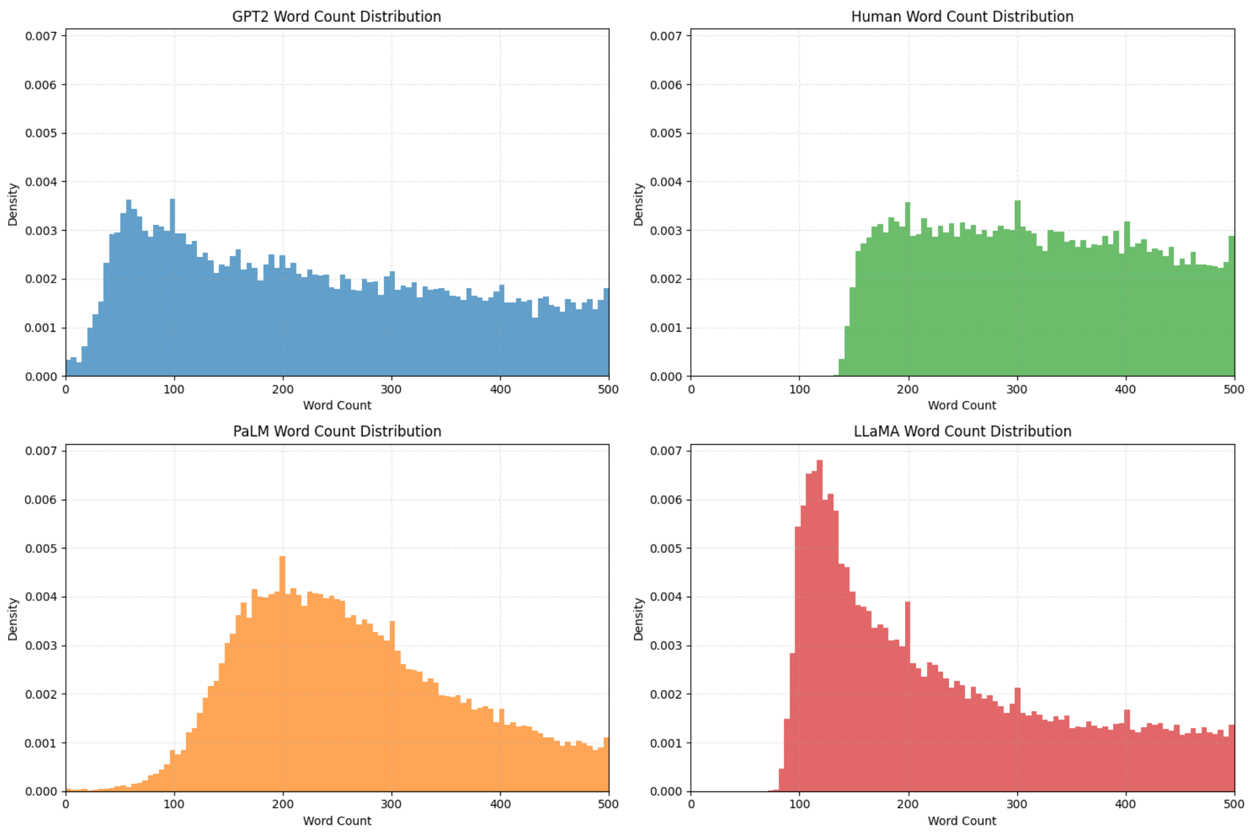

Fig. 3. Distribution of word counts in the multiclass OpenLLMText dataset.

V. Experiments and Results

This section will describe experiments aimed at achieving the detection of AI-generated text (binary classification experiments) and identifying the source model used for generation (multiclass classification experiments).

A. Experimental Setup

All experiments were conducted using **Google Colab Pro+**, equipped with an **NVIDIA A100 GPU**.

System specifications:

- **RAM:** 83 GB
- **GPU Memory:** 40 GB
- **Disk Space:** 200 GB

Training Time:

- **LSTM and BiLSTM models** were trained in approximately 60–80 min, with an average of 3 min per epoch.
- **BERT and RoBERTa models** required significantly more computation, taking 20–30 min per epoch, with total training times ranging from 1 h to 1 h and 40 min.

Libraries and Frameworks:

- **Data preprocessing and handling:** `datasets`, `pandas`, `nltk`, `numpy`, `scikit-learn`
- **Embedding generation:** `gensim` (for Word2Vec embeddings)
- **Model development and training:** `tensorflow.keras`, `PyTorch`

These computational resources and software tools provided a robust and scalable environment for conducting experiments involving the binary classification and multiclass source identification tasks.

B. Binary Classification Experiments

For the binary classification experiments, we utilized the GPT-Wiki-Intro dataset, which contains human-generated and LLM-generated text. Although the full dataset includes approximately 100,000 samples, we used a balanced subset of 40,000 samples for training, which is comprised of 20,000 human-generated and 20,000 AI-generated texts. Additionally, to ensure a fair comparison, we equalized the number of words in each sample by removing extra sentences from longer records.

Several experiments were conducted as part of this stage: an initial experiment to determine the effect of data preprocessing on this task, and three primary experiments evaluating LSTM's performance at binary classification using different embedding techniques.

1) Initial experiment——Effect of preprocessing: For the initial experiment, the necessity of preprocessing was investigated by evaluating its impact on the performance of two representative models:

- **LSTM**: A traditional recurrent neural network model.
- **BERT Classifier**: A pretrained transformer-based language model.

The objective was to determine whether preprocessing (e.g., tokenization, stop-word removal, lemmatization) improves the detection of AI-generated text in the GPT-Wiki-Intro dataset. These experiments did not include hyperparameter tuning or cross-validation, as the focus was solely on evaluating the effect of preprocessing.

The results of the experiments are summarized in Table IV. Accuracy, precision, recall, and F1-score were measured for each model, both with and without preprocessing.

TABLE IV: Performance Metrics for LSTM and BERT with and without Preprocessing

Model	Preprocessing	Accuracy	Precision	Recall	F1-Score
LSTM	Without	0.89	0.89	0.90	0.89
LSTM	With	0.79	0.75	0.88	0.81
BERT	Without	0.90	0.89	0.91	0.91
BERT	With	0.82	0.80	0.83	0.81

Based on these results, it is evident that both LSTM and BERT performed better without preprocessing. For each metric, the score was higher when the model was trained using the non-preprocessed dataset. For example, LSTM experienced a performance drop of approximately 9% in F1-score with preprocessing, and BERT saw a performance drop of approximately 11% in F1-score with preprocessing.

The decrease in performance when preprocessing was used may be attributed to the loss of contextual and stylistic features that are crucial for distinguishing between human-written and AI-generated text.

Based on these results, we conclude that preprocessing was counterproductive in AI-generated text detection. Therefore, we conducted all subsequent experiments without data preprocessing.

2) LSTM experiments with different embeddings: For the next experiments, we evaluated the performance of the LSTM model using different embedding techniques: one-hot embeddings, Word2Vec, and BERT embeddings. For each experiment, we performed hyperparameter tuning and cross-validation to ensure a robust evaluation. The performance was measured using accuracy, precision, recall, and F1-score. Additionally, the best hyperparameters that were identified during tuning are reported for each embedding method.

a) LSTM with one-hot embeddings: We first trained the LSTM model using one-hot embeddings and tuned the model hyperparameters to improve performance.

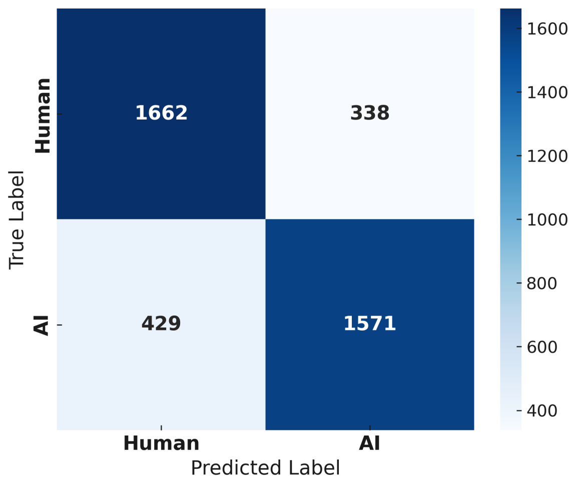

Fig. 4. Confusion matrix for binary classification of AI-generated text using LSTM with one-hot embeddings.

The final best-performing hyperparameters for this experiment were:

- **Vocabulary Size:** 10,000
- **Input Dimension:** 10,000
- **Hidden Dimension:** 128
- **Number of Layers:** 1
- **Dropout Rate:** 0.15
- **Batch Size:** 64
- **Learning Rate:** 0.001
- **Optimizer:** Adam
- **Number of Epochs:** 32

When using one-hot embeddings, the LSTM model achieved decent performance, as it scored around 75% for all four metrics. While the model captured basic patterns in the text, the confusion matrix in Fig. 4 shows that it still struggled to separate human-written from AI-generated content, with many false positive and false negative classifications. These results highlight the limitations of one-hot representations and serve as a baseline for comparing more advanced embeddings in the next experiments.

b) LSTM with Word2Vec embeddings: We also trained the LSTM model using Word2Vec embeddings, and tuned the model hyperparameters to improve performance.

The final best-performing hyperparameters for this experiment were:

- **Hidden Dimension:** 128
- **Dropout Rate:** 0.3
- **Batch Size:** 32
- **Learning Rate:** 0.001
- **Loss Function:** Binary Cross-Entropy Loss (BCELoss)
- **Optimizer:** Adam
- **Number of Epochs:** 42

When using Word2Vec embeddings, the LSTM model achieved a better performance than the previous experiment using one-hot embeddings, with an accuracy of 87% and an F1-score of 89%. As shown in the confusion matrix in Fig. 5, the model was especially good at identifying AI-generated text with very few false negative misclassifications. However, it was less accurate when identifying human-written text, as there were a significant number of false positives where the model labeled a human-written text as AI. This indicates a bias towards predicting that a text is AI-generated. Nonetheless, these results still demonstrate the effectiveness of Word2Vec at capturing richer semantic patterns for improved binary classification.

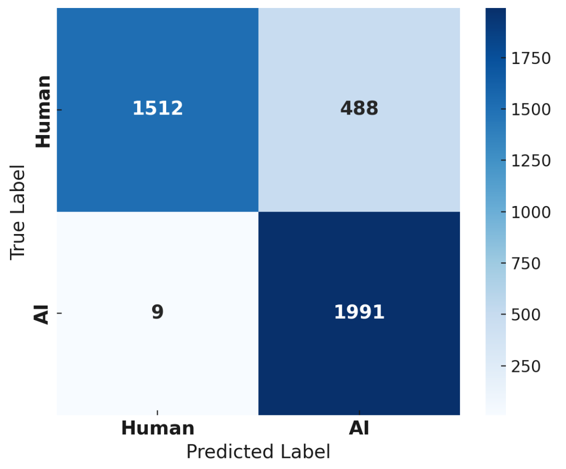

Fig. 5. Confusion matrix for binary classification of AI-generated text using LSTM with Word2Vec embeddings.

c) LSTM with BERT embeddings: For the third experiment, we trained the LSTM model using BERT (bert-base-uncased) embeddings, and tuned the model hyperparameters to improve performance. The final best-performing hyperparameters for this experiment were:

- **Embedding Size:** 100
- **Hidden Dimension:** 64
- **Dropout Rate:** 0.2
- **Batch Size:** 64
- **Learning Rate:** 0.001
- **Optimizer:** Adam
- **Number of Epochs:** 34

The LSTM model with BERT embeddings achieved 97% accuracy and an F1-score of 97%. As shown in the confusion matrix in Fig. 6, the model was highly accurate at identifying both human-written and AI-generated text, with very few misclassifications. This result highlights the power of contextual embeddings from BERT for capturing the subtle differences in language that simpler embedding methods might miss.

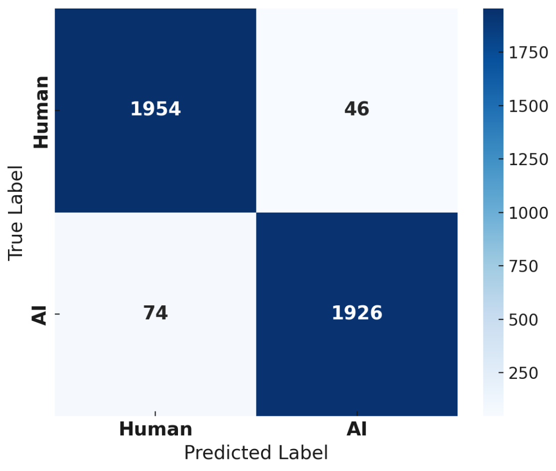

Fig. 6. Confusion matrix for binary classification of AI-generated text using LSTM with BERT embeddings.

d) Comparative analysis of embedding techniques for binary classification: To summarize, we compared the performance of the LSTM model using three different embedding techniques, as shown in Table V. Among the techniques, BERT embeddings delivered the highest overall performance across all metrics, followed by Word2Vec, which produced average results, then one-hot embeddings, which showed the weakest results.

TABLE V: Comparison of Performance Metric Scores for LSTM using Different Embedding Techniques for Binary Classification

Embedding Type	Accuracy	Precision	Recall	F1-Score
One-Hot	0.74	0.76	0.73	0.75
Word2Vec	0.87	0.80	0.99	0.89
BERT	0.97	0.94	0.96	0.97

These results indicate that the choice of embedding method significantly impacts the performance of the LSTM model for this application. BERT embeddings outperformed the other techniques across all metrics, likely due to BERT's ability to capture deep contextual information. Word2Vec embeddings also performed well, but did not achieve as strong results due to a possible bias towards positive predictions. One-hot embeddings produced the lowest performance, reflecting their inability to capture deep semantic and contextual relationships. Additionally, the different hyperparameters used for each experiment shows that optimal configurations vary depending on the embedding technique, emphasizing the importance of individual fine-tuning for each experiment.

C. Multiclass Classification Experiments

For the multiclass classification experiments, we utilized the OpenLLMText dataset, which contains text generated by three models (GPT, PaLM, and LLaMA) and human-written text. The dataset includes a large number of samples; however, for training purposes, we used a subset of 80,000 samples for training, which is comprised of 20,000 samples from each category (GPT, PaLM, LLaMA, and Human). Additionally, to ensure a balanced comparison across all categories, we equalized the word count in each record by removing extra sentences to maintain the word count close to 200. The updated word count distribution for each category is visualized in Fig. 7.

Two types of models were experimented with as part of this stage: traditional RNN-based models (LSTM and BiLSTM) and transformer-based LLMs (BERT and RoBERTa). Each experiment evaluated the selected model's performance at multiclass classification of generative AI sources.

1) LSTM with Word2Vec embeddings: We trained the LSTM model using Word2Vec embeddings generated from the multiclass OpenLLMText dataset. We also performed hyperparameter tuning with a focus on embedding size, number of LSTM units, learning rate, batch size, and dropout rate, and evaluated the model's performance at the task of distinguishing the sample texts between the four classes: Human, GPT2, LLaMA, and PaLM. The final best-performing hyperparameters for this experiment were:

- **Embedding Size:** 100
- **Hidden Dimension:** 128
- **Dropout Rate:** 0.2
- **Batch Size:** 32
- **Learning Rate:** 0.001
- **Optimizer:** Adam
- **Number of Epochs:** 39

The LSTM model trained with Word2Vec embeddings achieved a score of 55% for both accuracy and F1 when evaluated on the multiclass classification task. As shown in the confusion matrix in Fig. 8, the model was fairly successful in identifying the text generated by LLaMA and PaLM, but struggled to distinguish between the Human and GPT-2 outputs, as it often misclassified one as the other. For example, a significant number of human-written samples (895) were incorrectly predicted as GPT-2, and a considerable number of GPT2-generated samples (387) were incorrectly predicted as Human. This confusion suggests that while Word2Vec embeddings are able to capture basic semantic patterns, they may not provide enough contextual depth to differentiate subtle stylistic features.

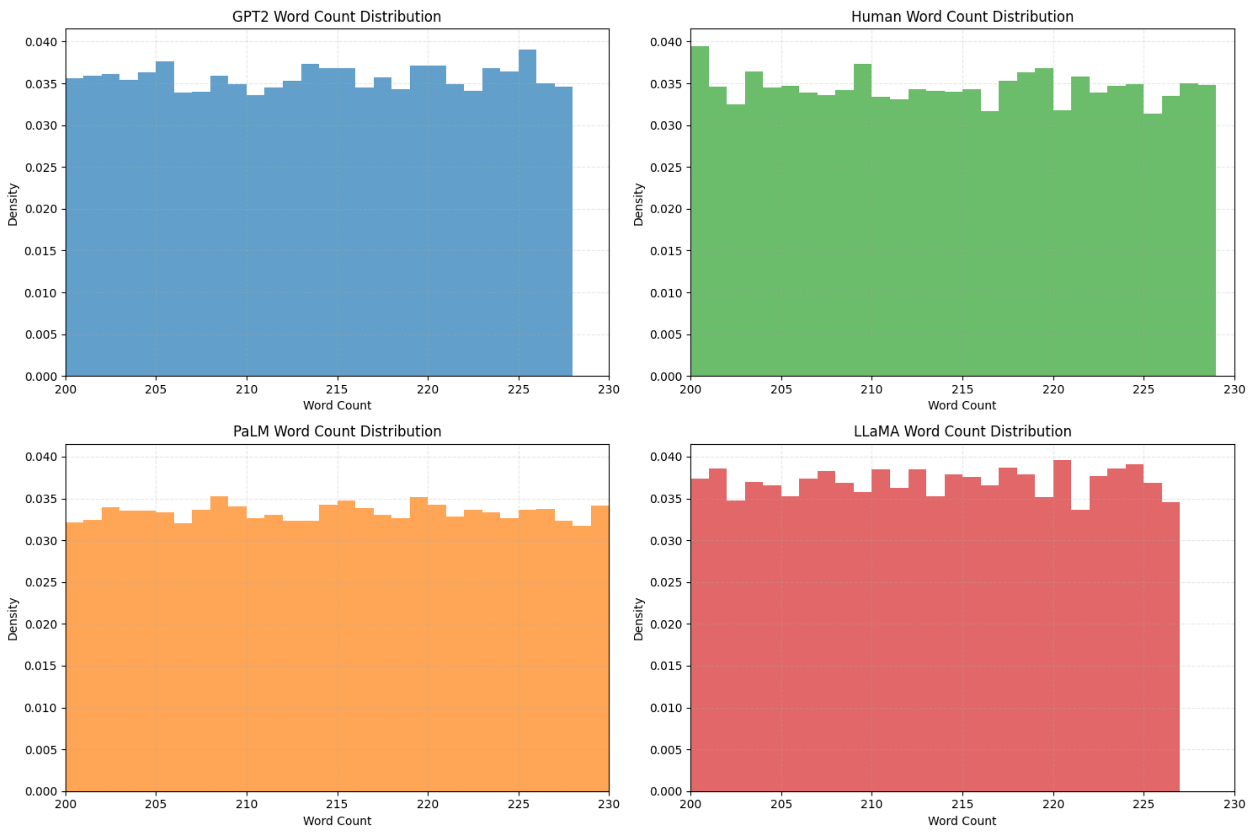

Fig. 7. Balanced distribution of word counts in the multiclass OpenLLMText dataset.

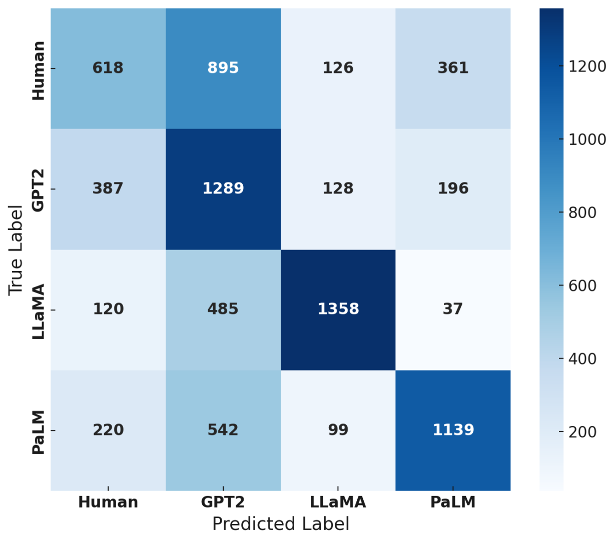

Fig. 8. Confusion matrix for multiclass classification of texts by different AI sources using LSTM with Word2Vec embeddings.

2) BiLSTM with Word2Vec embeddings: We trained the BiLSTM model using Word2Vec embeddings generated from the multiclass OpenLLMText dataset, and tuned the model hyperparameters to improve performance. The final best-performing hyperparameters for this experiment were:

- **Embedding Size:** 100
- **Hidden Dimension:** 128
- **Dropout Rate:** 0.3
- **Batch Size:** 32
- **Learning Rate:** 0.0005
- **Optimizer:** Adam
- **Number of Epochs:** 34

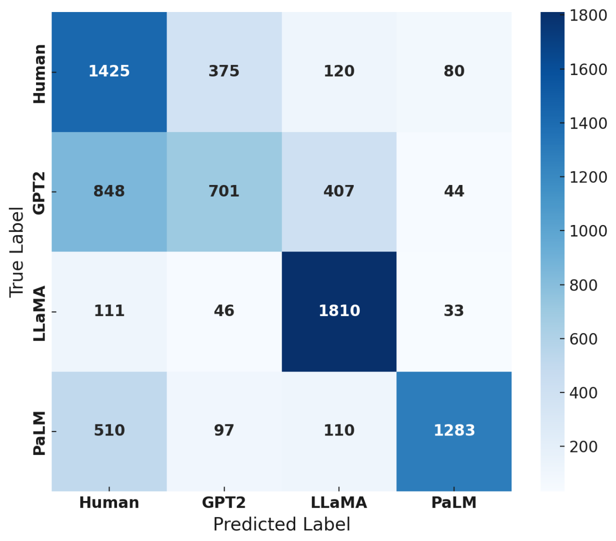

Fig. 9. Confusion matrix for multiclass classification of texts by different AI sources using BiLSTM with Word2Vec embeddings.

The BiLSTM model trained with Word2Vec embeddings achieved an accuracy of 57% and an F1-score of 56%. The confusion matrix in Fig. 9 shows that the model performed well on the LLaMA and PaLM classes, but struggled with GPT2. The model misclassified 848 GPT2 texts as Human and 407 GPT2 texts as LLaMA, with only 701 correct GPT2 classifications. Although the bidirectional structure helped improve classification of human-written samples, the static Word2Vec embeddings still limited the model's ability to capture fine-grained differences between sources.

3) BERT: For a different approach, we trained the transformer-based LLM BERT (bert-base-uncased) for this multiclass text classification task. We fed BERT the Open-LLMText dataset, and trained it while performing hyperparameter tuning to improve performance. The final best-performing hyperparameters for this experiment were:

- **Batch Size:** 8
- **Learning Rate:** 5e-5
- **Weight Decay:** 0.01
- **Number of Epochs:** 5
- **Loss Function:** Cross-Entropy Loss
- **Optimizer:** AdamW

The BERT model performed well on the multiclass classification task, achieving 87% for accuracy, precision, and recall, along with an F1-score of 86%. As shown in the confusion matrix in Figure 10, BERT accurately classified most samples across all four categories, with a particularly high accuracy for LLaMA and PaLM outputs. While BERT still had a few hundred misclassifications for Human and GPT2, it was a major improvement compared to the traditional LSTMs. The low number of misclassifications demonstrates BERT's effectiveness at capturing the distinct language patterns of each source.

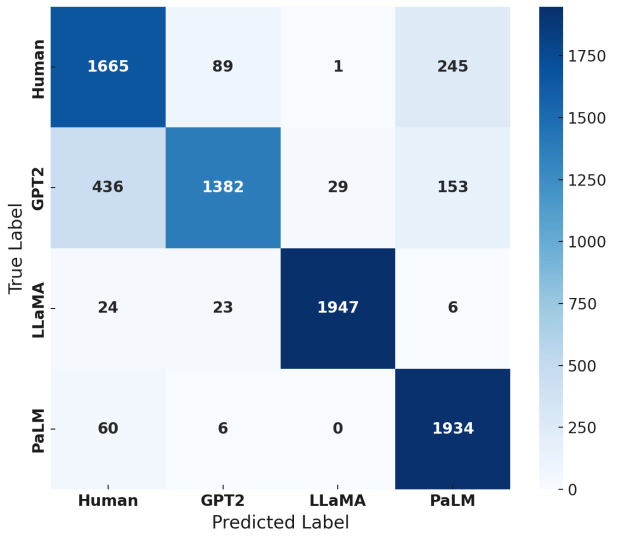

Fig. 10. Confusion matrix for multiclass classification of texts by different AI sources using BERT.

4) RoBERTa: We also trained RoBERTa (roberta-base-uncased), an optimized version of BERT, for multiclass classification. Like with BERT, we fed RoBERTa the Open-LLMText dataset, and trained it while performing hyperparameter tuning to improve performance. The final best-performing hyperparameters for this experiment were:

- **Base Model:** RoBERTa-base
- **Max Sequence Length:** 200
- **Batch Size:** 8
- **Learning Rate Scheduler:** Warmup Steps: 500
- **Weight Decay:** 0.01
- **Number of Epochs:** 4

The RoBERTa model performed slightly better than BERT on the multiclass classification task, achieving an accuracy of 88%, a precision of 89%, a recall of 88%, and an F1-score of 88%. The confusion matrix in Figure 11 reveals that the model classified most samples correctly across all four categories, with an especially high accuracy for LLaMA and PaLM texts. A few hundred GPT-2 and Human samples were misclassified, yet this still reflects RoBERTa's superior ability to capture nuanced differences in writing style across sources.

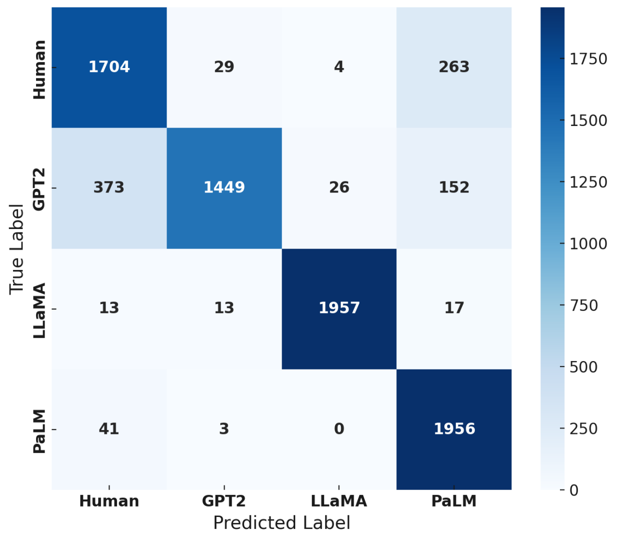

Fig. 11. Confusion matrix for multiclass classification of texts by different AI sources using RoBERTa.

5) Comparative analysis of multiclass classification models: As shown in Table VI, the performance metrics across different models reveal key insights into their effectiveness for the multiclass classification task of identifying the source behind a generated text.

TABLE VI: Performance Comparison and Analysis of Multiclass Classifiers

Model	Accuracy	Precision	Recall	F1
Word2Vec + LSTM	0.55	0.58	0.55	0.55
Word2Vec + BiLSTM	0.57	0.59	0.57	0.56
BERT	0.87	0.87	0.87	0.86
RoBERTa	0.88	0.89	0.88	0.88

Traditional RNN-based Models (LSTM and BiLSTM):

LSTM and BiLSTM, both using Word2Vec embeddings, yielded accuracies of 55% and 57%, respectively. The BiLSTM model showed a modest improvement over LSTM, likely due to its ability to process input in both forward and backward directions, yet it was not a sufficient enhancement. These results highlight the limitations of RNN-based architectures when tasked with capturing complex contextual relationships in multiclass textual data.

Transformer-based Models (BERT and RoBERTa):

The transformer-based models outperformed the RNN-based models by a large margin. BERT achieved an accuracy of 87%, benefiting from its self-attention mechanism that enables a deeper understanding of context. RoBERTa slightly surpassed BERT with an accuracy of 88%, as well as higher scores for precision, recall, and F1. This subtle improvement can be attributed to RoBERTa's more extensive pretraining, larger model size, and better-optimized hyperparameters during fine-tuning.

VI. Results Analysis and Discussion

The experimental results underscore the critical role of embedding techniques and model architectures in the performance of text classification tasks, particularly in distinguishing AI-generated content and identifying its source model.

For the binary classification task, the LSTM model combined with BERT embeddings significantly outperformed its counterpart configurations, which used Word2Vec and one-hot embedding methods. The superior performance of BERT embeddings (accuracy and F1-scores of 97%) can be attributed to BERT's ability to capture deep contextual and semantic relationships within text. In contrast, LSTM with Word2Vec achieved a moderately high performance (accuracy of 87% and F1-score of 89%). This configuration benefited from learned word relationships but lacked the contextual depth of transformer-based representations. One-hot embeddings, which lack semantic richness and context-awareness, delivered the weakest performance (accuracy of 74% and F1-score of 75%). These findings highlight the importance of leveraging contextualized embeddings for tasks involving nuanced distinctions such as human vs. AI-generated content.

In the multiclass classification setting, the difference between traditional RNN-based models and transformer-based models is even more pronounced. LSTM and BiLSTM models, both using Word2Vec embeddings, yielded relatively low accuracies (55% and 57%, respectively), revealing their deep limitations when handling complex and subtle multiclass scenarios. The slight advantage of BiLSTM over LSTM suggests some benefit from bidirectional context, but this gain was minimal.

On the other hand, transformer-based LLMs, particularly BERT and RoBERTa, delivered significantly better results, with RoBERTa achieving the highest accuracy and F1-Score (both 88%). This improvement stems from transformer models' attention mechanisms and large-scale pretraining, which enable richer contextual understanding across long sequences. RoBERTa's edge over BERT likely results from its enhanced pretraining strategy and architectural optimizations.

Overall, model architecture and input representation are both decisive factors in AI-generated text detection and source identification. Traditional RNNs, even with quality embeddings, are insufficient for fine-grained classification. In contrast, transformer models not only excel in accuracy but also offer more robust generalization capabilities, making them well suited for real-world deployment in AI content detection systems.

VII. Conclusion and Future Work

This research focused on detecting AI-generated text in two settings: binary classification (AI vs. Human) and source model identification (Human, PaLM, LLaMA, GPT2). As predicted, RNN-based models performed well at binary classification. The best-performing configuration was LSTM trained using BERT embeddings, which achieved accuracy and F1-scores of 97%. However, identifying the source of AI-generated text proved to be a more challenging task. The best performing model was the RoBERTa LLM, with an accuracy of 88%, which suggests that its enhanced pretraining strategies are well-suited for handling diverse and nuanced datasets. Our findings indicate that transformer models are highly suitable for AI-generated text detection tasks, and offer superior performance across various standard evaluation metrics. This research also lays the groundwork for further advancements in AI-generated text identification, paving the way for real-world applications of more scalable and robust models.

For our future work, we plan to explore dataset diversification by including data from various other domains, such as news articles, social media, and academic papers, as well as incorporating multilingual sources to enhance the generalizability and robustness of the proposed models. Also, we strive to investigate optimization techniques such as model quantization, pruning, Low-Rank Adaptation (LoRA), and efficient inference frameworks (e.g., ONNX or TensorRT) to optimize the models for real-time deployment scenarios, such as live content moderation systems.

This research holds significant potential for real-world applications. As generative AI continues to advance, our framework can assist in identifying AI-generated content in fields such as education and journalism, where maintaining authenticity is essential. Our technology may also support plagiarism detection in academic and professional settings. Furthermore, our source model identification system will help increase the accountability of LLMs.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

A.P.T. performed the experiments and drafted the manuscript. F.A. supervised the project, guided the methodology and analysis, validated results, revised the manuscript, and addressed reviewers' comments. N.A. contributed to experimental design refinement, result validation, and manuscript revisions. All authors reviewed and approved the final version.

Fundings

This research was supported by the U.S. National Science Foundation (Award#: 2319803). Opinions, findings, and conclusions are those of the authors and do not necessarily reflect the views of the NSF.

Acknowledgment

We would like to thank Jade Webb and Tien Nguyen from the Department of Computer Science at San José State University for their invaluable contributions to the final revisions and proofreading of this manuscript.

References

[1] J. A. S. de Cerqueira, M. Agbese, R. Rousi, N. Xi, J. Hamari, and P. Abrahamsson, "Can we trust ai agents? an experimental study towards trustworthy llm-based multi-agent systems for ai ethics," 2024. [Online]. Available: https://arxiv.org/abs/2411.08881

[2] L. F. Gutiérrez, F. Abri, M. Armstrong, A. S. Namin, and K. S. Jones, "Email embeddings for phishing detection," in *2020 IEEE International Conference on Big Data (Big Data)*, 2020, pp. 2087–2092.

[3] F. Abri, L. F. Gutiérrez, A. S. Namin, K. S. Jones, and D. R. W. Sears, “Linguistic features for detecting fake reviews,” in *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2020, pp. 352–359.

[4] T. Nguyen, F. Abri, A. S. Namin, and K. S. Jones, “Deception and lie detection using reduced linguistic features, deep models and large language models for transcribed data,” in *2024 IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*, 2024, pp. 376–381.

[5] Y. Tsubota and Y. Kano, “Text generation indistinguishable from target person by prompting few examples using LLM,” in *Proceedings of the 2nd International AIWolfDial Workshop*, Y. Kano, Ed. Tokyo, Japan: Association for Computational Linguistics, Sep. 2024, pp. 13–20. [Online]. Available: https://aclanthology.org/2024.aiwolfdial-1.2

[6] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, “Improving language understanding by generative pre-training,” *OpenAI blog*, 2018, https://openai.com/research/language-unsupervised.

[7] H. Touvron, T. Lavril, G. Izacard *et al.*, “Llama: Open and efficient foundation language models,” *arXiv preprint arXiv:2302.13971*, 2023.

[8] G. DeepMind, “Gemini technical report,” 2023, https://deepmind.google/technologies/gemini.

[9] OpenAI, “Gpt-4 technical report,” 2024. [Online]. Available: https://arxiv.org/abs/2303.08774

[10] A. Dubey, “The llama 3 herd of models,” 2024. [Online]. Available: https://arxiv.org/abs/2407.21783

[11] H. McGovern, R. Stureborg, Y. Suhara, and D. Alikaniotis, “Your large language models are leaving fingerprints,” 2024. [Online]. Available: https://arxiv.org/abs/2405.14057

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” 2019. [Online]. Available: https://arxiv.org/abs/1810.04805

[13] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, “Roberta: A robustly optimized bert pretraining approach,” 2019. [Online]. Available: https://arxiv.org/abs/1907.11692

[14] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[15] A. Graves and J. Schmidhuber, “Framewise phoneme classification with bidirectional lstm networks,” in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 4, 2005, pp. 2047–2052 vol. 4.

[16] R. Shijaku and E. Canhasi, “Chatgpt generated text detection,” *Publisher: Unpublished*, 2023.

[17] J. Luo, G. Nan, D. Li, and Y. Tan, “Ai-generated review detection,” *Available at SSRN 4610727*, 2023.

[18] T. B. Brown, B. Mann, N. Ryder, M. Subbiah *et al.*, “Language models are few-shot learners,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.

[19] R. Corizzo and S. Leal-Arenas, “One-class learning for ai-generated essay detection,” *Applied Sciences*, vol. 13, no. 13, p. 7901, 2023.

[20] G. Bao, Y. Zhao, Z. Teng, L. Yang, and Y. Zhang, “Fast-detectgpt: Efficient zero-shot detection of machine-generated text via conditional probability curvature,” *arXiv preprint arXiv:2310.05130*, 2023.

[21] Z. Deng, H. Gao, Y. Miao, and H. Zhang, “Efficient detection of llm-generated texts with a bayesian surrogate model,” *arXiv preprint arXiv:2305.16617*, 2023.

[22] A. Radford, J. Wu, R. Child *et al.*, “Language models are unsupervised multitask learners,” *OpenAI blog*, 2019, https://openai.com/research/language-unsupervised.

[23] H. Touvron *et al.*, “Llama 2: Open foundation and fine-tuned chat models,” *arXiv preprint arXiv:2307.09288*, 2023.

[24] A. Bhattacharjee, T. Kumarage, R. Moraffah, and H. Liu, “Conda: Contrastive domain adaptation for ai-generated text detection,” *arXiv preprint arXiv:2309.03992*, 2023.

[25] R. Gaggar, A. Bhagchandani, and H. Oza, “Machine-generated text detection using deep learning,” 2023.

[26] L. Mindner, T. Schlippe, and K. Schaaff, “Classification of human- and ai-generated texts: Investigating features for chatgpt,” in *International Conference on Artificial Intelligence in Education Technology*. Springer, 2023, pp. 152–170.

[27] D. Nguyen, K. M. N. Naing, and A. Joshi, “Stacking the odds: Transformer-based ensemble for ai-generated text detection,” *arXiv preprint arXiv:2310.18906*, 2023.

[28] Y. Wang, J. Mansurov, P. Ivanov, J. Su, A. Shelmanov, A. Tsvigun, C. Whitehouse, O. M. Afzal, T. Mahmoud, A. F. Aji, and P. Nakov, “M4: Multi-generator, multi-domain, and multi-lingual black-box machine-generated text detection,” 2023.

[29] Y. Chen, H. Kang, V. Zhai, L. Li, R. Singh, and B. Raj, “Token prediction as implicit classification to identify llm-generated text,” *arXiv preprint arXiv:2311.08723*, 2023.

[30] T. Fagni, F. Falchi, M. Gambini, A. Martella, and M. Tesconi, “Tweepfake: About detecting deepfake tweets,” *Plos one*, vol. 16, no. 5, p. e0251415, 2021.

[31] K. Wu, L. Pang, H. Shen, X. Cheng, and T.-S. Chua, “Llmdet: A third party large language models generated text detection tool,” *arXiv preprint arXiv:2305.15004*, 2023.

[32] N. Maloyan, B. Nutfullin, and E. Ilyshin, “Dialog-22 ruatd generated text detection,” in *Computational Linguistics and Intellectual Technologies*. RSUH, Jun. 2022. [Online]. Available: http://dx.doi.org/10.28995/2075-7182-2022-21-394-401

[33] K. Hayawi, S. Shahriar, and S. S. Mathew, “The imitation game: Detecting human and ai-generated texts in the era of chatgpt and bard,” 2023. [Online]. Available: https://arxiv.org/abs/2307.12166

[34] A. Chowdhery, S. Narang, J. Devlin *et al.*, “Palm: Scaling language modeling with pathways,” *arXiv preprint arXiv:2204.02311*, 2022.

[35] Aaditya Bhat, “Gpt-wiki-intro (revision 0e458f5),” 2023. [Online]. Available: https://huggingface.co/datasets/aadityaubhat/GPT-wiki-intro

[36] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *arXiv preprint arXiv:1301.3781*, 2013.

[37] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.

[38] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, “Roberta: A robustly optimized bert pretraining approach,” *arXiv preprint arXiv:1907.11692*, 2019.