Detecting Human-Written and AI-Generated English Text Using NLP

Dr. Soha Ahmed
Computer Science Department
Faculty of Computers and Artificial Intelligence, Helwan University
Cairo, Egypt

***Abstract*— The rapid emergence of large language models (LLMs) such as ChatGPT, Bard, and Claude has increasingly blurred the line between human-authored and AI-generated text. In this paper, we present a comparative study of deep learning and classical machine learning approaches for binary text classification: determining whether English text is written by a human or generated by an AI model. We use different methods, including semantic embeddings from a fine-tuned BERT transformer, statistical patterns captured via TF-IDF, and topic distributions derived from Latent Dirichlet Allocation (LDA). These features are evaluated individually and in hybrid configurations using classifiers such as Logistic Regression, Support Vector Machines (SVC), and Random Forests. Experiments are performed on a balanced dataset with over 100,000 samples. Among all methods tested, the BERT-based classifier achieved the highest validation accuracy of 97.5% and exceeded the performance of traditional pipelines. These results demonstrate the ability of transformers to extract more depth of linguistic patterns and provide a strong baseline for future research in AI-generated text provenance detection.**

Keywords—AI-generated text, Human-written text, Text classification, BERT, Deep learning, Machine learning, Text detection, TF-IDF, LDA

I. INTRODUCTION

The swift uptake of generative AI models such as ChatGPT, GPT-3.5, and GPT-4 has generated an immediate necessity to distinguish between English text that has been written by humans and that which is machine-generated. The models can produce accurate grammar and contextually appropriate material that seems to be virtually indistinguishable from the writing of humans. This has profound implications across a broad variety of industries from education to journalism, publishing, and digital ethics [8], [11].

In the last ten years, there have been various approaches proposed to address the task of AI-text detection. Among these are rule-based systems and statistical classifiers, progressing to more advanced deep learning systems. Traditional machine learning techniques, such as Term Frequency–Inverse Document Frequency (TF-IDF) [7], [9], and Latent Dirichlet Allocation (LDA) topic modeling, have been moderately effective in identifying linguistic features characteristic of AI-generated text. Still, more current deep learning architectures—specifically transformer-based models such as Bidirectional Encoder Representations from Transformers (BERT) [3], [5], have been able to outperform on account of their power to learn rich semantic and contextual representations of language [1], [4].

This paper presents a robust system for detection that combines classical machine learning approaches with deep learning models for detecting AI-generated English text. The system is experimented using a compound dataset that has been created from three publicly accessible datasets: DAIGT [15], LLM-Detect-AI [14], and HC3 [13]. A single preprocessing and feature engineering pipeline is run on the combined dataset, and then various models are trained and evaluated. Experimental results show that the model based on BERT has the best classification accuracy, thereby proving its effectiveness in detecting machine-generated content [5].

The key contributions of this work are as follows. First, it presents a comparative model study of deep learning and traditional models for the task of AI-text detection [2], [3]. Second, it shows the value of the heterogeneity combination of datasets to achieve better generalization of models and offset dataset-specific biases [4], [6]. Third, the study presents a detailed comparison with the state-of-the-art methods such as TF-IDF with SVM and LDA with Random Forest classifiers which demonstrate that the BERT model outperforms [3], [5], [7]. Finally, a novel hybrid detection model is introduced, employing semantic, statistical, and structural characteristics to further enhance detection [11].

The remainder of this paper is structured as follows. Section II gives an overview of the literature review. Section III explains the proposed detection methodology. Section IV covers the experimental setup and results. Section V concludes the research and sets out directions for future work.

II. LITERATURE REVIEW

AI-generated content has also become an area of growing study with the development of advanced language models such as GPT-3.5 and GPT-4. Various approaches have been proposed to tackle the problem, ranging from commercial applications to university research models, employing various means and gaining varying degrees of accuracy [1], [4].

Commercial tools like Copyleaks and GPTZero have been developed to detect AI-generated content. Copyleaks relies on a proprietary model with training based on known AI-generated text to identify stylistic patterns, whereas GPTZero relies on statistical metrics such as perplexity and burstiness for identifying machine-generated text [6], [11]. Although the tools are simple to use and accessible, they tend to be opaque overall and work less well for processing content produced by newer and more advanced models. Besides, they may provide incorrect results when applied to short or paraphrased content [3].

Deep learning algorithms, on the other hand, have provided more promising outcomes in the research field. The transformer-based models such as BERT, RoBERTa, and ELECTRA have been particularly fine-tuned for AI-text detection usage [5], [10]. The deep contextual relations between text can be caught by these models, and they have the ability to observe subtle differences between human and artificial writing. BERT, particularly, has performed well in the task when trained on different datasets, and thus it is among the highest-performing models for this task. However, these models are computationally costly and require a huge amount of labeled data, hence limiting their scalability and accessibility [4].

Alternatively, traditional machine learning models are still valuable since they are easy to interpret and simple. Techniques using TF-IDF with classifiers such as Support Vector Machines or Logistic Regression are surface-level statistical properties of the text-based. Topic modeling methods such as Latent Dirichlet Allocation (LDA) with Random Forest classifiers also aim to describe thematic inconsistencies present in text written by AI [7], [9]. While these models are more computationally efficient and easier to deploy, they often do not possess the semantic depth needed to match transformer-based models in terms of accuracy and robustness.

In addition to single models, hybrid models have been proposed to merge semantic embeddings with structural and statistical properties. These models try to integrate the benefits of conventional methods and deep approaches, which can be balanced in performance most of the time. However, the success of these models highly depends on careful feature selection and tuning [11], [12].

Despite these advances, existing methods have several weaknesses. The majority of them fail to detect text that has been edited or paraphrased by a human after generation from an origin AI. Many are data-specific and fail to generalize across domains or styles of writing. There is also the issue of lack of interpretability of deep learning models, which will make it hard for them to apply practically in sensitive applications such as education or journalism.

These challenges underscore the ongoing need for robust, adaptive, and open detection models that are capable of keeping pace with shifting capability from generative AI.

III. PROPOSED METHODOLOGY

This section explains the datasets used, preprocessing pipeline, feature extraction techniques, classification models, and implementation for our AI-created text detection system A comprehensive overview of the full system—covering all four models presented in this study, including TF-IDF with linear classifiers, LDA with Random Forest, BERT-based classification, and the hybrid feature engineering pipeline—is shown in Figure 1.

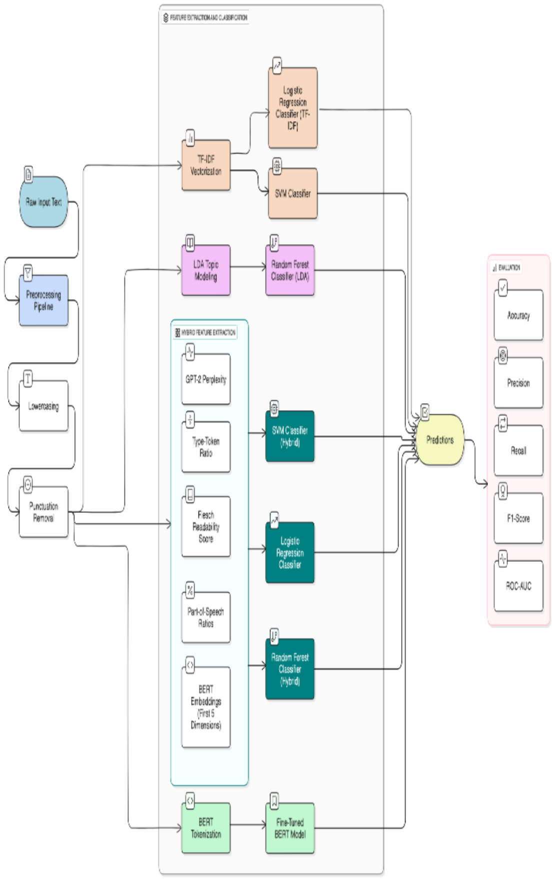


Figure 1: Framework for all proposed models.

A. Data Description

We utilized three publicly available datasets: DAIGT [15], LLM-Detect-AI[14], and HC3[13], all of which contain labeled examples of human-written and AI-generated English text [4], [6], [10]. These datasets were merged into a single corpus, after which duplicate entries and overly short samples were removed. Each record in the unified dataset consists of two columns: text, containing the raw English paragraph or sentence, and label, a binary value indicating the source — 0 for human-written and 1 for AI-generated. To balance the class, we randomly sampled an equal proportion of human-written and machine-generated texts to end up with a final dataset of 106,912 samples, with 53,456 labeled as machine-generated and 53,456 as human-written. We then stratified the dataset into training, validation, and test splits with ratios 80%, 10%, and 10%.

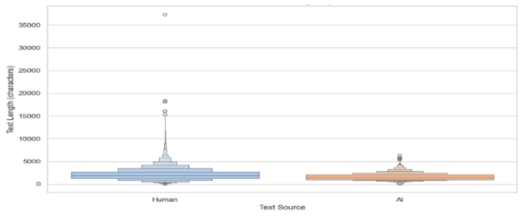


Figure 2. Box-and-line plots.

To further investigate differences in text structure we visualized some of the key statistical measures using box-and-line plots. As depicted in Figure 2, human-created text is generally much more varied in terms of sentence length and lexical diversity relative to AI-generated text, which appears to be more clustered around higher uniformity in structure.

B. Pre-Processing

All text samples underwent the same preprocessing pipeline for uniformity and to eliminate noise from the dataset. Typical preprocessing includes lowering case, eliminating punctuation, digits, special characters, and extra white-space. For traditional machine learning models (e.g., LDA and TF-IDF pipelines) stopwords are removed to assist in term discrimination and reduced noise. In addition to stopword removal, the processes for tokenization were done using Count Vectorizer or TF-IDF Vectorizer, respectively [7], [9].

In contrast, transformer-based models (like BERT), use raw text with little preprocessing. For example, stopwords and punctuation are left as is, since BERT uses the WordPiece tokenizer for subwords and contextual meaning [5]. Preprocessing modifications for each method will be described in their respective sections below.

C. Implementation Details

Method 1: Topic Modeling with LDA and Random Forest Classification

This method uses a classical topic modeling approach in conjunction with an ensemble classifier for detecting AI-generated text. First, the original text data is passed through a preprocessing pipeline that converts the content to lowercase, removes punctuation, digits, and special characters, and removes common English stop-words to help eliminate noise and increase discrimination. After cleaning and readying the text, each document was converted to a document-term matrix using a Count Vectorizer that was configured to extract unigrams and bigrams while restricting to the top 8,000 most frequent tokens.

Next, we applied a Latent Dirichlet Allocation (LDA) as an unsupervised generative probabilistic framework to find latent semantic topics in the text corpus. In using LDA, we assume that each document is a mixture of many topics, and that each topic can be explained by a probability distribution over words. For variables covered by the LDA generative process, we use Dirichlet priors on document's topic distributions and topic's word distributions, estimating latent posterior distributions through variational inference. Notably, LDA produces a low-dimensional representation of each document as a vector of topic probabilities, capturing the underlying thematic structure that may differentiate AI-generated from human-written content. The generative process for a document d with N words is formally defined as:

$$\begin{aligned}
\theta_d &\sim \mathrm{Dir}(\alpha) \quad \text{(document-level topic distribution)}\\
z_{dn} &\sim \mathrm{Multinomial}(\theta_d) \quad \text{(topic assignment for each word)}\\
w_{dn} &\sim \mathrm{Multinomial}(\phi_{z_{dn}}) \quad \text{(word distribution for selected topic)}
\end{aligned}$$

Figure 3 shows that LDA describes each document as a mixture of topics, and each topic as a distribution over words. It allows for LDA to find latent thematic structures that will help us separate AI-generated text from untainted, human-generated text. The 30-dimensional topic vectors, [7], were then used as input

lom Forest classifier with 300 trees and a maximum depth of 30. The Random Forest was used for its robustness, ability to model complex decision boundaries, and the ability to model the structure without extensive hyper-parameter tuning [2]. The 80% of the balanced dataset was used to train the model while the remaining consisted of 20% of data for validation and evaluation. The model achieved an overall accuracy of 94%, demonstrating the Random Forest's ability to accurately discriminate between AI and human texts based on topic distributions. Examination of the top words per topic yielded meaningful semantic clusters that mirror the characteristic differences in writing style and how content was generated [9].

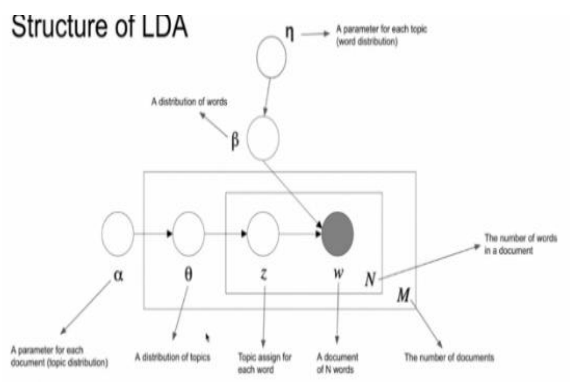

Structure of LDA
η
A parameter for each topic (word distribution)
A distribution of words
β
α
θ
z
w
N
M
The number of words in a document
A parameter for each document (topic distribution)
A distribution of topics
Topic assign for each word
A document of N words
The number of documents

Figure 3. Graphical Model of LDA

Method 2: TF-IDF Vectorization with Linear Classification Models

In this method, We use a classical feature extraction method—Term Frequency-Inverse Document Frequency (TF-IDF)—combined with linear classifiers to separate AI generated text from human-written text. The method focuses on extracting discriminative lexical patterns from textual data, while benefiting from the advantages of sparse feature representation and efficient decision boundaries [3].

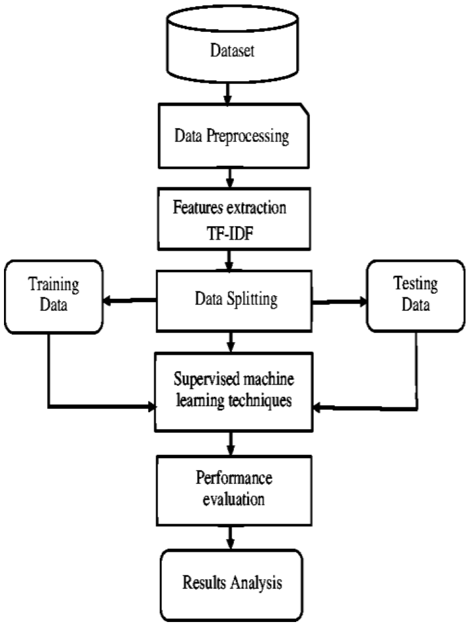


Figure 4. Pipeline for TF-IDF

Figure 4 shows the TF-IDF pipeline used for feature extraction. The pipeline is initialized by performing standard text preprocessing to normalize and clean the raw input. This means we convert text to lowercase, remove punctuation, digits, special characters, and eliminate English stopwords. Performing these actions helps reduce lexical noise, ensuring that learned features are reflecting semantically-relevant content, and are not a by-product of formatting and grammar artifacts.

After pre-processing, the cleaned text is transformed into a vector using the TF-IDF method, which transforms each document into a high-dimensional sparse feature vector representing the importance of each term with respect to the full corpus. Term frequency (TF), describes how often a term occurs in a document, while inverse document frequency (IDF) serves to lessen the weight of relatively common terms that appear in many documents. This two-component weighting process allows common, non-informative terms to score more poorly while highlighting unique terms with uniquely contextual significance. Formally, for a term t contained in document d, its TF-IDF score is defined as:

$$\boldsymbol{TF(t,d)} = \frac{(Number\ of\ occurrences\ of\ term\ t\ in\ document\ d)}{(Total\ number\ of\ terms\ in\ the\ document\ d)}$$

$$\boldsymbol{IDF(t,D)} = \log_e \frac{(Total\ number\ of\ documents\ in\ the\ corpus)}{(Number\ of\ documents\ with\ term\ t\ in\ them)}$$

$$TF\text{-}IDF(t,d,D) = TF(t,d) \times IDF(t,D)$$

The TF-IDF vectors are then fed into two linear classifiers: Logistic Regression (LR) and Linear Support Vector Machine (SVM). Logistic Regression estimates the posterior probability of class membership as a sigmoid function of the linear combination of input features, using L2 regularization to control overfitting and numerical instability. Linear SVM finds the hyperplane that maximizes the margin between the two classifications in a high-dimensional TF-IDF space. It is applicable with high-dimensional, sparse text data and is controlled by an L1 penalty term with squared hinge loss; thus enforcing sparsity and robustness. Figure 3 illustrates the TF-IDF pipeline used for feature extraction. Text is pre-processed and passed as a sparse matrix to linear classifiers, such as SVM or Logistic Regression for classification.

The dataset is randomly split into an 80% training and 20% testing subset, while adhering to stratified sampling [7], [9] to ensure class balance. Both classifiers are trained on the TF-IDF features from the training set and evaluated on the test set with standard performance metrics. The Logistic Regression model produced a classification accuracy of approximately 91.6%. The Linear SVM classification provides a performance boost, achieving a classification accuracy of 95.9% [2], [3]. These results help demonstrate the effectiveness of scaling TF-IDF representations with linear classifiers for effective and interpretable text classification tasks.

Method3: BERT-Based Classification

This research employs BERT (Bidirectional Encoder Representations from Transformers) as the primary model to classify a text as AI-generated or Human-written. BERT is a pretrained deep-learning model that is based on the Transformer encoder architecture introduced by Vaswani et al. and adapted by Devlin et al. BERT is different from typical language models because it learns using a bidirectional context, enabling it to more accurately capture context from words based on text on both the left and right side of the word in a sentence. Having context in both directions, BERT can better model relationships between words and understand the semantics better than unidirectional models [5], [10].

The Transformer encoder forms the basis of BERT, which builds multiple identical layers with a self-attention mechanism and feed forward neural network. The self-attention mechanism creates relations between every word pair in the input sequence by projecting the tokens to query, key, and value vectors, and then calculates the weighted attention scores. This permits the model to emphasize relevant words, independent of their location within the input sequence. BERT uses multi-head attention as well. This component divides the attention operation into a number of parallel, smaller subspaces, which improves the model's ability to capture multiple linguistic traits. Each layer has residual connections for the attention score and feedforward neural network outcome, and layer normalized output to stabilize during training and improve performance. Figure 5 shows the BERT model architecture.

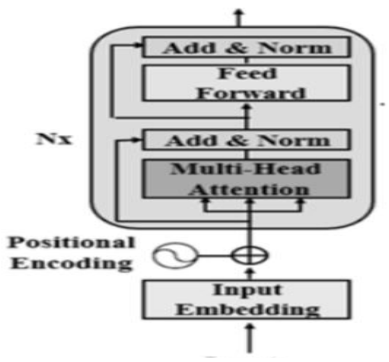


Figure 5: Architecture of (BERT).

BERT is pre-trained on large-scale unlabeled corpora using two self-supervised tasks: Masked Language Model (MLM) and Next Sentence Prediction (NSP). MLM masks random parts of the input text and trains the model to predict the masked words which allows for deep contextual representations to be learned. NSP trains BERT to predict if a sentence follows another and promotes the learned component that supports understanding the relationship between sentences. Then, BERT can be fine tuned on a downstream task with relatively small labeled datasets by adding an output layer that is task-specific , Figure 6 illustrates the end-to- end training process of the BERT model.

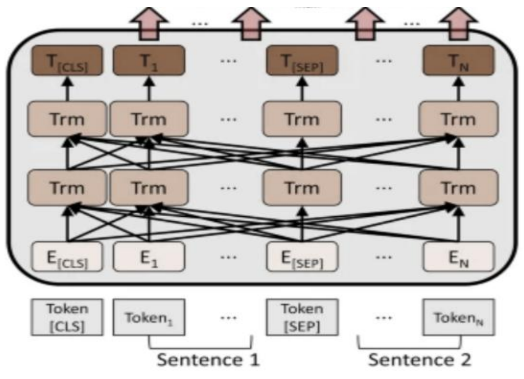
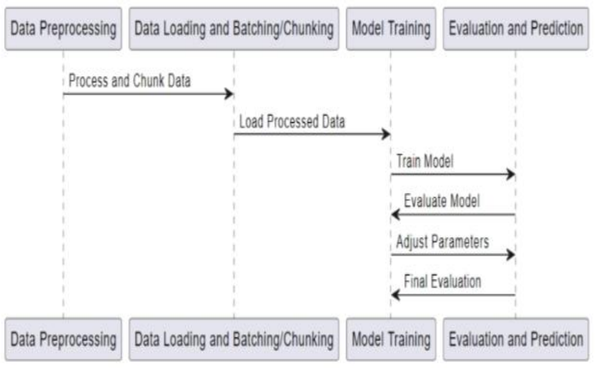


Figure 6: The training process of the BERT model.



Prior to being fine-tuned, preprocessed raw text data is subjected to a formal preprocessing phase to standardize as much as possible and ensure good quality input. The preprocessing steps include changing the text to lowercase, and stripping punctuation, digits, HTML entities, and non-ascii characters, as well as normalizing whitespace. Unlike traditional pipelines, stopwords are preserved in the text, as BERT's contextual embeddings benefit more from the full linguistic structure. After the data has been cleaned, the text can be tokenized using BERT's WordPiece tokenizer, which outputs token IDs for each sentence in the form of a sequence. The tokenizer also creates attention masks that indicate which tokens in the sequence are real tokens and which are padding, and produces segment IDs if there are more than one sentence present. To ensure that all inputs to the BERT pipeline are 256 tokens long (the maximum length allowed), all sequences must be padded or removed to fit the boundary [5]. To train the model, batches will be constructed “on-the-fly” within a custom generator; this is necessary because the entire dataset resides in CSV files with more than five million records. The ability to read in chunks allows the model to not breathe in memory with large datasets, and processing can occur in real-time while the model trains. For the fine-tuning, we will use the AdamWeightDecay optimizer and a sparse categorical cross-entropy loss function. The model will be trained for 3 epochs, using both training accuracy and validation accuracy to monitor model performance.

The BERT model fine-tuned for our task achieved a very good classification performance, obtaining a training accuracy of 97.12% and validation accuracy of 97.50%. The results emphasize BERT's capable abilities to differentiate AI-produced and human-written text, based on its ability to employ deep bidirectional attention and strong pre-trained language understanding [4], [5].

Method 4: Hybrid Feature Engineering and Classical Machine Learning Classification

Besides the deep learning models, we designed a hybrid feature engineering pipeline that accompanies classical machine learning classifiers for the task of differentiating AI-generated text from human-written text. This pipeline encompasses purely linguistic as well as semantic features extracted from the text. These features are subsequently input into three classical classifiers: Random Forest, Logistic Regression, and Support Vector Machine (SVC).

The preprocessing step included the cleaning of the input text by nullifying the non-alphanumeric characters and converting everything into lowercase to have uniformity. Incomplete records with missing text were eliminated to ensure consistency and integrity in the process of feature extraction. A predefined set of hand-crafted features was calculated to characterize each sample of the text after preprocessing. The features involved are perplexity, burstiness, type-token ratio (TTR), Flesch Reading Ease score, part-of-speech (POS) ratios, and a diminished set of BERT embeddings [11].

Perplexity refers to the measurement of how well a probability distribution predicts a given set of data. Perplexity is meant to show in which text the language model predicts better; usually, a lower perplexity score stands for better fluency and coherence. To capture structural variability, which is often greater in human-written text, we applied the burstiness feature, defined as the standard deviation of sentence lengths relative to their mean. TTR serves as one measure of lexical diversity: it calculates that ratio using unique words in the numerator and total words in the denominator. The Flesch Reading Ease score gives an estimate of how easy or difficult it is for someone to understand a piece of text and was measured here using the textstat library. For grammatical analysis, we calculated the relative frequencies of nouns, verbs, and adjectives using part-of-speech tagging because those grammatical elements vary in proportion between AI-generated and human-authored texts.

We also extracted semantic information using BERT embeddings for the sake of complementing these linguistic features. Specifically, we took the mean of the last hidden state of BERT and retained the first five dimensions as dense numerical features. These features were then appended to the dataset alongside the linguistic attributes. We cleaned the resulting feature matrix by converting infinities to NaNs and dropping rows with incomplete data; more specifically, we addressed infinite or missing values in the feature matrix.

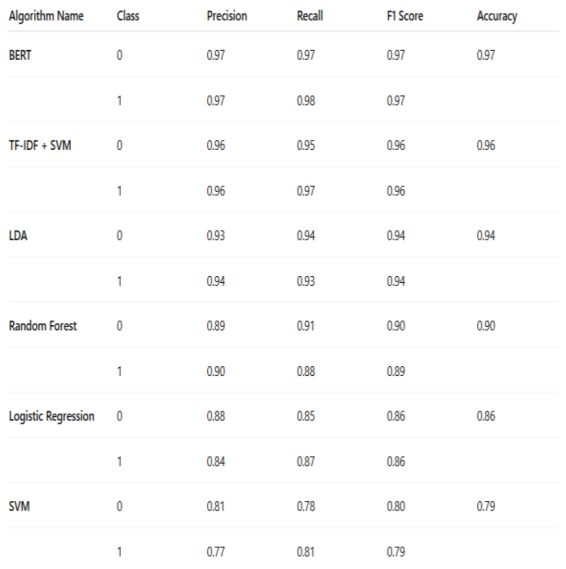
A final set of features trained the three classifiers. For the Random Forest classifier, which is an ensemble technique based on multiple decision trees, 200 estimators were configured, and a fixed random seed for reproducibility. In Logistic Regression, the binary classification linear model, maximum convergence iterations were at 1000 with parallel processing and a fixed random state [7]. Lastly, probability estimates were enabled for improving interpretability while keeping the seed in Support Vector Machine.

Of the three models, Random Forest gave the best performance with an accuracy of 89.56, followed by Logistic Regression and SVC. The results thus corroborate that, upon careful engineering, linguistic and semantic features are capable of effectively distinguishing human from AI-generated content even in the context of traditional machine learning approaches [11], [12].

IV. EXPERIMENTS AND RESULTS

In this section, we describe the experiment and share the results of testing several classifiers on the task of classifying AI-generated text versus human-generated text. We used a range of models from traditional machine learning techniques to deep learning models based on transformers. Our assessment of the performance of individual models, either using metrics such as precision, recall, or F1-score, or overall accuracy.

TABLE I. PERFORMANCES OF DIFFERENT CLASSIFIERS



Algorithm Name	Class	Precision	Recall	F1 Score	Accuracy
BERT	0	0.97	0.97	0.97	0.97
	1	0.97	0.98	0.97	
TF-IDF + SVM	0	0.96	0.95	0.96	0.96
	1	0.96	0.97	0.96	
LDA	0	0.93	0.94	0.94	0.94
	1	0.94	0.93	0.94	
Random Forest	0	0.89	0.91	0.90	0.90
	1	0.90	0.88	0.89	
Logistic Regression	0	0.88	0.85	0.86	0.86
	1	0.84	0.87	0.86	
SVM	0	0.81	0.78	0.80	0.79
	1	0.77	0.81	0.79	

Table 1 presents the classification performance of all models. From the evaluation of six different machine learning models for detecting text generated by artificial intelligence (AI), BERT achieved the best accuracy of 97%. This concurs with the fact that BERT is a state-of-the-art transformer-based model that can capture complex contextual relationships in a text [3], [5]. TF-IDF combined with SVM follows closely with an accuracy of 96%, while the Latent Dirichlet Allocation (LDA) model performed solidly at 94% [7], [9].

As for traditional machine learning classifiers, Random Forest exhibited good performance (90%) compared to Logistic Regression (86%) and SVM (79%) [1], [10]. These classical models do a good job, however, they typically trail transformer-based and embedding-dense methods due to their limited ability to capture semantic subtleties and complicated linguistic patterns [11], [12].

In conclusion, BERT's heightened accuracy suggests its robustness in discerning subtle linguistic cues that are used to differentiate AI-generated text and human-generated text, while traditional classifiers still yield reasonably reliable results and are computationally inexpensive, transformer-based models like BERT are recommended when an application requires substantial precision and recall.

V. CONCLUSION

This paper presented a large-scale comparative analysis of classic machine learning techniques and recent deep learning models towards the identification of AI-generated and human-generated English text. Based on a single and balanced dataset from multiple benchmarks [4], [6], we used and compared four methodologies, from LDA with Random Forest to TF-IDF with linear models, and a transformer model developed using BERT, along with a hybrid feature engineering pipeline that includes linguistic and semantic features.

Our experiments reveal that transformer-based models, particularly BERT, significantly outperform conventional methods, with a maximum validation accuracy of 97.5% [3], [5]. This underscores the power of contextual semantic modeling in recognizing fine-grained patterns and linguistic structures responsible for differentiating AI from human language. While conventional models such as TF-IDF with SVM and LDA with Random Forest performed well and consume less computation [7], [9], they are not capable of recognizing the deeper contextual semantics in complex structures of language .

The hybrid model incorporating statistical, structural, and semantic characteristics proved to be a useful middle ground that holds promise for the integration of classical classifiers with modern feature embeddings. Explorations of explainable AI techniques will be pursued in future work to more fully tackle interpretability, as well as robustness in cross-domain generalization and adversarial scenarios involving paraphrased or edited AI text [11], [12]. Lastly, this work establishes a solid foundation for continuing with reliable and scalable solutions for the ubiquitous task of AI text detection.

VI. FUTURE WORK

Although the system in this study performed well, several ways exist that could improve it for future research to make it more useful and widely applicable.

A better improvement is to support different languages and topics. The system currently supports only English. There are possibilities for development in other languages, like Arabic, Hindi, or Chinese, especially those with other scripts and grammatical structures. Including texts in different domains and informal styles would also enhance the performance of the system based on actual circumstances.

Another challenge is detecting short texts, such as social media posts or headlines, which give less context for the model. This could be improved by using smaller, faster models like DistilBERT or new attention-based techniques made for short content.

Explainability is also essential. SHAP or LIME can be used in future versions to show which words or features influenced the model's decision. This would render the system more comprehensible and reliable to users, especially in areas sensitive in nature, such as education or online investigations.

With newer models like ChatGPT, Claude, and Gemini still pushing the boundaries, it may also be helpful to know which model generated the text. Future systems may have features or classifiers specifically designed to detect the writing style of specific models.

Finally, future work must have ethics and legality in mind. This involves making the system fair for all languages, avoiding mistakes that identify human writing as AI, and protecting individuals' privacy.

Generally, this research is a good base, and future research can make AI-text detection more flexible, interpretable, and applicable to more languages and scenarios.

REFERENCES

[1] G. H. de Vries, “Can AI-generated texts be detected? A critical evaluation of detection techniques,” arXiv preprint, arXiv:2304.04736, Apr. 2023.

[2] K. Grgić, “Challenges of distinguishing AI-generated from human-written texts,” Open Information Science, vol. 6, no. 1, pp. 203–213, Jan. 2022, doi: 10.1515/opis-2022-0158.

[3] T. A. Pan, L. Dai, Y. Liu, and S. Qian, “LLM-detector: An ensemble-based framework for detecting text generated by large language models,” arXiv preprint, arXiv:2501.03212, Jan. 2025.

[4] C. Sun, Y. Zhang, and X. Liu, “A benchmark for detecting AI-generated text: Large-scale dataset and evaluation,” arXiv preprint, arXiv:2405.16422, May 2024.

[5] A. Gao, M. S. Sap, and D. Bamman, “Human or machine? Detecting machine-generated text from GPT,” arXiv preprint, arXiv:2301.07597, Jan. 2023.

[6] B. Terry, “Evaluating the efficacy of AI content detection tools in differentiating between human and AI-generated text,” ResearchGate, Aug. 2023.

[7] M. I. Afli, M. U. S. Khan, and A. Al-Ahmari, “Automatic detection of machine-generated text: A critical survey,” ResearchGate, Oct. 2020.

[8] N. Jain and R. Khurana, “Academic integrity in the age of generative AI: Detecting ChatGPT-written content in education,” Interactive Technology and Smart Education, vol. 21, no. 1, pp. 101–115, Mar. 2024, doi: 10.1108/ijilt-03-2023-0043.

[9] S. Ju, L. Qin, and J. Wang, “Detecting machine-generated academic text using stylometric analysis,” in Proc. Int. Conf. on Applications of Natural Language to Information Systems (NLDB), Salford, UK, Jun. 2016, pp. 391–396, doi: 10.1007/978-3-319-41754-7_43.

[10] J. Ko and S. Kim, “DetectGPT2: Unsupervised detection of AI-generated text via local perturbation,” arXiv preprint, arXiv:2404.10032, Apr. 2024.

[11] J. Berić, “How to detect AI-generated texts,” ResearchGate, Nov. 2023.

[12] E. Solaiman, M. Brundage, L. Clark, A. Askell, M. Krueger, and G. Irving, “Release strategies and the social impacts of language models,” arXiv preprint, arXiv:2011.01314, Nov. 2020.

[13] Z. Solotko, R. Jones, and H. SimpleAI, "HC3: A dataset for detecting AI-generated responses," Hugging Face Datasets, 2023.

[14] S. Thite, "LLM-Detect-AI: Dataset for detecting AI-generated student essays," Kaggle, 2024.

[15] A. Paullier, "DAIGT v2: Dataset of AI-generated and human-written essays," Kaggle, 2023.