# Detection of Text Generated by ChatGPT

Mohammad Asghar
*Student Number: 160546576*
Professor Arkaitz Zubaiga
*MSc. Data Science and Artificial Intelligence*

***Abstract*— The proliferation of Large Language Models (LLMs) like ChatGPT has significantly blurred the line between human-written and AI-generated text, raising concerns about academic integrity and content authenticity. This project explores the detection of AI-generated text by evaluating the performance of three models: a Random Forest classifier, an LSTM neural network, and a BERT-based transformer. Using the 'aadityaubhat/GPT-wiki-intro' dataset, these models were trained and tested to identify text generated by ChatGPT. The study finds that the BERT-based transformer model outperforms the others, achieving a test accuracy of 97.50% and a perfect ROC-AUC score of 1.00, making it the most effective tool for distinguishing AI-generated content. This research underscores the critical need for advanced detection techniques to maintain the integrity of digital communications in the era of AI.**

***Keywords*— *AI-generated text, ChatGPT, Random Forest, LSTM neural network, BERT-based transformer, Large Language Models, academic integrity, content authenticity.***

## I. Introduction

In recent years, the rise of Artificial Intelligence (AI) tools, particularly Large Language Models (LLMs), has marked a significant milestone in technological advancement. With the increasing availability of vast datasets and rapid enhancements in computing power, AI has evolved into a powerful tool capable of addressing complex problems, making accurate predictions, and offering valuable insights across various domains and industries (Aggarwal et al., 2022). Initially, chatbots were restricted to answering brief, straightforward questions and participating in domain-specific conversations. However, the development of transformer-based models such as GPT and BERT has revolutionized this landscape. Modern chatbots now possess the capability to process intricate, lengthy queries and generate detailed responses in real-time, excelling in a wide range of text types including emails, recipes, poetry, meeting summaries, essays, and even algorithms or code (Oghaz et al., 2023).

As these models continue to grow in sophistication, the distinction between human and AI-generated text is becoming increasingly blurred. AI-generated text, also known as machine- or computer-generated text, is produced through the application of natural language processing (NLP) and machine learning (ML) technologies. With the ongoing advancements in AI, this technology has become remarkably proficient at crafting original, high-quality essays. If not effectively monitored and detected, this development could pose a significant threat to academic integrity and the authenticity of student work (Cingillioglu, 2023). Recently, numerous journals have revised their publication guidelines, generally prohibiting ChatGPT from being listed as an author and mandating full disclosure of its use (Flanagin et al., 2023).

The excessive reliance on AI-generated text could result in an influx of manuscripts with limited value, potentially leading to an over-representation of highly cited papers, while newer, less recognized work might be overlooked (Grimaldi and Ehrler, 2023). A significant concern is the propensity of these tools to "hallucinate," creating false information that doesn't correspond to reality (Azamfirei, Kudchadkar, and Fackler, 2023).

Therefore, as more systems like ChatGPT become increasingly integrated into our daily lives, distinguishing between content created by humans and that generated by AI becomes crucial. While both types of content can convey information, the key difference lies in the intent behind the text. Human-generated content is crafted with a specific purpose in mind, whereas AI-generated content is produced by algorithms designed to mimic human writing. AI-generated text may exhibit repetitive or formulaic patterns, whereas human-generated text tends to be more original and creative. Additionally, text produced by LLMs often sounds convincing, even though it is based on word probabilities rather than factual accuracy (Mindner, Schlippe, and Schaaff, 2023).

Numerous tools, such as GPTZero and the GPT-2 Output Detector, utilize methods like evaluating text complexity and variation to detect AI-generated content. Despite these measures, such tools often face challenges with accuracy, frequently resulting in false positives where human-written content is incorrectly labeled as machine-generated. These inaccuracies can lead to ethical concerns and damage to reputations (Mindner, Schlippe, and Schaaff, 2023).

This report delves into the recent developments in AI text detection technologies, with a focus on identifying outputs from models like ChatGPT. It will assess the performance of various tools, highlight their limitations, and explore the consequences of their application in different professional and academic environments. This analysis aims to emphasize the critical need for continual advancements in this field to ensure that progress in AI-Text generation is matched by our ability to detect and mitigate its effects, thus preserving the integrity of digital communications.

## II. Related Work (Literature Review)

In this section, recent milestones and significant advancements in text generation by LLMs, specifically ChatGPT, will be examined. The discussion begins with the 'Characteristics of ChatGPT-Generated Text,' detailing how the model processes and generates language. This is followed by an overview of 'Existing Methods for AI Text Detection,' emphasizing the effectiveness and challenges of current techniques. Lastly, the 'Challenges in Distinguishing ChatGPT Output from Human Writing' are explored, focusing on error rates and detection accuracy.

### *A. Characteristics of ChatGPT-Generated Text*

The characteristics and features of ChatGPT-generated text have become an important focus of research, especially as AI-produced content becomes more prevalent in academic and professional contexts. A notable study by **Alamleh et al. (2023)** emphasized the exceptional coherence and fluency of ChatGPT-generated text, which frequently makes it challenging to distinguish from human-written content. Their study found that classical machine learning models, such as Random Forest (RF) and Support Vector Machine (SVM), achieved high accuracy rates of 92.5% and 91.0%, respectively, in identifying AI-generated text. In contrast, deep learning models like BERT showed lower accuracy (69.7%), primarily due to the small dataset size of 500 data points, which favors classical algorithms like RF and SVM suitable (Alamleh et al., 2023).

Another significant characteristic of ChatGPT-generated text is its tendency to produce shorter paragraphs, less variation in sentence length, and a more formulaic use of punctuation. In contrast, human writing typically involves longer paragraphs, a wider range of sentence structures, and more subtle language use. **Desaire et al. (2023)** explored this in their study by developing an XGBoost classifier model to distinguish ChatGPT-generated content from human-written text. They trained this model on a dataset comprising 192 documents, including 1,276 paragraphs for training and 1,210 paragraphs for testing. The model achieved over 99% accuracy in distinguishing ChatGPT-generated science writing from human-authored texts. Key features used in the model included paragraph complexity, sentence-level diversity, punctuation usage, and specific word choices (Desaire et al., 2023).

Moreover, **Fröhling and Zubiaga (2021)** introduced a feature-based classifier for distinguishing between human and machine-generated text, addressing concerns about the potential misuse of advanced language models. Their approach utilizes carefully crafted features to model intrinsic differences in linguistic characteristics, likely including aspects of syntax, lexical diversity, and repetitiveness. The classifier demonstrated competitive performance compared to more complex and computationally expensive methods, particularly in generalizing across different sizes of the same language model. Focusing on state-of-the-art models like GPT-2, GPT-3, and Grover, the study revealed that different sampling methods in text generation lead to distinct types of flaws, which the classifier could detect. This feature-based approach offers an accessible and effective "first line-of-defense" against the abuse of language models, achieving high effectiveness in detecting machine-generated text without requiring extensive computational resources (Fröhling and Zubiaga., 2021)

### *B. Existing Methods for AI Text Detection*

The rapid advancement of AI-generated text has spurred the development of various detection methods. These approaches can be broadly categorized into statistical methods, machine learning techniques, and hybrid approaches that combine multiple strategies. In this context, the study by **Liang et al. (2023)** provides a critical examination of existing GPT detectors, particularly against non-native English writers. The research involved testing seven widely-used GPT detectors on a dataset comprising 91 TOEFL essays from non-native writers and 88 US 8th-grade essays from native speakers. The findings revealed a significant bias, with over 61.22% of the non-native essays being misclassified as AI-generated, while native essays were accurately identified. To mitigate this bias, the researchers enhanced the linguistic diversity of non-native samples using ChatGPT, which reduced the false positive rate to approximately 11.77%. This intervention increased the text perplexity of the essays, aligning them more closely with native writing samples (Liang et al., 2023).

**Yang et al. (2023)** also contributed to the field of AI text detection by introducing the "Polish Ratio," a novel metric designed to quantify the extent of AI involvement in modifying human-written texts. This research addressed the growing challenge of distinguishing between human-written and AI-modified content, particularly focusing on texts polished by ChatGPT. The study presented the Human-ChatGPT Polished Paired Texts (HPPT) dataset, which includes over 6,050 pairs of original and ChatGPT-polished abstracts, emphasizing subtle AI-induced changes. The Polish Ratio employed similarity metrics such as Jaccard and Levenshtein distances to measure textual modifications, providing a continuum from 0 (no modification) to 1 (entirely modified). Experimental results demonstrated that models utilizing the Polish Ratio achieve a 94.65% accuracy on the HPPT dataset and significantly outperformed traditional models on more complex datasets like ChatGPT-Detector-Bias-Dataset (CDB), which contained native and non-native English texts (Yang et al., 2023). These advancements have important implications for maintaining text integrity in fields like academia and journalism.

In exploring existing methods for AI text detection, the study by **Mitchell et al. (2023)** offers a significant advancement who introduced "DetectGPT," a novel method that discerns text generated by LLMs using an innovative approach based on the curvature of the model's log probability function. This method identified a key characteristic of LLM-generated text—it tends to occupy areas of negative curvature within the model's log probability landscape, distinguishing it from human-generated text which does not exhibit consistent probabilistic patterns upon perturbations. By evaluating the log probabilities of a text and its perturbed versions, where perturbations are minor modifications using another pre-trained model like T5, DetectGPT effectively determined the likelihood of the text being machine-generated. This approach eliminated the need for training specific classifiers or compiling targeted datasets, significantly improving detection capabilities. Demonstrated in their study, DetectGPT elevated the detection accuracy for fake news articles generated by the 20-billion parameter GPT-NeoX from a 0.81 to a 0.95 AUROC, marking a substantial improvement over previous zero-shot methods and illustrating its potential utility in fields where verifying the authenticity of text is paramount (Mitchell et al., 2023).

### *C. Challenges in Distinguishing ChatGPT Output from Human Writing*

The task of differentiating between ChatGPT-generated text and human writing presents numerous complex challenges. The study conducted by **Bommasani et al. (2021)** provided critical insights into the capabilities and risks associated with large-scale AI models like GPT-3, BERT, and CLIP. These foundation models, particularly GPT-3 with its 175 billion parameters, have demonstrated advanced linguistic abilities, generating text with such fluency and

coherence that it is often indistinguishable from human writing at first glance. Notably, in specific tasks, non-experts misclassified GPT-3 generated text as human-written, highlighting the growing difficulty in detecting AI-generated content. This aligned with the model's performance in tasks such as the NY Regents 8th grade science exam, where adapted models achieved a 91.6% accuracy, further complicating the differentiation between AI and human outputs (Bommasani et al., 2022).

In a significant exploration of the challenges inherent in distinguishing AI-generated content from human writing, **Rashidi et al. (2023)** conducted a study published in the *Journal of Pathology Informatics*, assessing the reliability of AI text detection tools. They analyzed 14,400 abstracts from a variety of esteemed journals such as Nature and Science, spanning from 1980 to 2023. The findings illuminated considerable issues with current detection models, revealing that 5% to 10% of authentic human-generated abstracts were incorrectly identified as AI-generated. Notably, The New England Journal of Medicine exhibited the highest misidentification rate at 10.24%. This study underscores the critical limitations and challenges faced by detection technologies, emphasizing the urgent need for advancements to ensure accurate differentiation and prevent the potential mislabeling of scholarly work, which could have serious professional consequences for authors (Rashidi et al., 2023).

Moreover, **Dergaa et al. (2023)** carried out an in-depth analysis providing insight into the profound capabilities and inherent risks associated with AI in academic settings. Their study, which employed a quasi-qualitative analysis of peer-reviewed articles indexed in Scopus, underscored that AI tools like ChatGPT can efficiently automate academic writing tasks, producing content nearly indistinguishable from human-generated work. However, they raised significant concerns regarding the authenticity of such outputs, citing a case where only 63% of fake AI-generated abstracts were identified by experts, highlighting a critical vulnerability in the peer review process. This points to the pressing need for robust mechanisms to ensure transparency and uphold the integrity of academic publications in the era of advanced AI technologies (Dergaa et al., 2023).

## III. METHODOLOGY

This section details the methodology employed in the project to identify text produced by ChatGPT. The process is structured into several key stages, each essential to achieving the desired outcomes. These stages encompass Dataset Overview and Preprocessing, the application of three distinct models (a Classical Machine Learning Model, a Neural Network Architecture, and a Deep Learning Model), and the evaluation of these models. Each subsection offers a comprehensive explanation of the techniques and procedures utilized to accomplish the study's goals.

**Note:** The models were created using Google Colab.

### *A. Dataset Overview and Preprocessing*

For the study, the '**aadityaubhat/GPT-wiki-intro**' dataset was chosen for its extensive collection of paired human-written and GPT-generated Wikipedia introductions. This dataset, with over 150,000 entries, is ideal for training models to detect AI-generated text, providing both content and metadata such as the number of words in each introduction and the prompt used to generate the AI text. This structured data facilitates in-depth analysis and model training for the detection task making it ideal for this project (Bhat, 2023).

#### *1) Data Preprocessing*

##### *a) Data Labelling:*

To facilitate the training of models to differentiate between human-written and machine-generated texts, the dataset required clear labeling, which was done by assigning a label of **0** to all **human-written introductions** and a label of **1** to **machine-generated introductions**. These labeled texts were then combined into a single column, creating a structured and balanced dataset.

##### *b) Data Balancing:*

To confirm that the models did not become biased towards either class (human-written or machine-generated), the dataset was balanced. The steps included in terms of balancing the dataset were: (1). Identifying the size of the smaller class, and (2). Randomly sampling the larger class to match this size, ensuring an equal number of entries for both classes.

##### *c) Dataset Sampling and Splitting:*

After balancing the dataset, the steps taken to prepare it for training, validation, and testing were: (1). Randomly selecting a 20% subset of the balanced data for the entire process, and (2). Splitting this subset into three parts—70% for training, 15% for validation, and 15% for testing. This approach ensured that the models were trained, validated, and tested on distinct portions of the data, allowing for an accurate assessment of their generalization capabilities.

### *B. Model Implementation and Architecture*

This section outlines the implementation and architecture of three models employed in this study to differentiate between human-authored and ChatGPT-generated text. The study explores a spectrum of methodologies, ranging from traditional machine learning to sophisticated deep learning techniques. The models examined include a Random Forest classifier representing classical machine learning, a neural network based on Long Short-Term Memory (LSTM) for deep learning, and a transformer model based on BERT, also within the deep learning category. Each model incorporates specific preprocessing, data augmentation, and fine-tuning strategies, which are elaborated upon in the following sections (Oghaz et al., 2023).

#### *1) Model 1: Random Forest Classifier with TF-IDF Vectorization (Classical Machine Learning Approach)*

##### *a) Preprocessing and Tokenization in NLP*

Text preprocessing and tokenization are crucial steps in Natural Language Processing (NLP). Preprocessing involves cleaning raw text to enhance data quality and model performance by removing noise and inconsistencies. Tokenization breaks the text into smaller units, such as words or sentences, allowing for structured analysis. Together, these processes prepare textual data for effective machine learning applications (GeeksforGeeks, 2024).

##### *b) TF-IDF Vectorization*

TF-IDF (Term Frequency-Inverse Document Frequency) is a vectorization method that converts text into numerical representations. It highlights the significance of words based on their frequency in a document relative to the entire dataset. (Karabiber, 2024).

***c) Application in the Model***

For this study, TF-IDF vectorization was applied to the dataset to prepare the text for input into the Random Forest classifier. The '**TfidfVectorizer**' from the scikit-learn library was utilized, with parameters set to capture both unigrams and bigrams (n-gram range of 1-2) (Scikit-learn developers, 2024). The maximum number of features was limited to 10,000 to ensure that the most relevant terms were included while avoiding overfitting. Stop words—common words that do not contribute significant meaning—were excluded from the vectorization process to reduce noise and improve the model's focus on meaningful terms (BotPenguin, 2024).

***d) Random Forest (RF) Classifier***

The Random Forest classifier is an ensemble technique that constructs numerous decision trees on different subsets of the data and then averages their predictions. This approach reduces the risk of overfitting and is renowned for its robustness and accuracy, making it well-suited for managing complex, high-dimensional datasets like those found in text classification (Breiman, 2001).

***e) RF Classifier Implementation***

In this project, the Random Forest classifier was implemented using the '**RandomForestClassifier**' class from the scikit-learn library (Scikit-learn developers, 2024). The model's hyperparameters were fine-tuned through a grid search, a process that systematically tests different combinations of parameter values to find the most effective configuration for the specific dataset (Selvaraj, 2022). The hyperparameters optimized included (Koehrsen, 2018); **n_estimators** (Number of trees in the forest, set to 300), **max_depth** (Maximum depth of the trees was set to None), **min_samples_split (**Minimum number of samples required to split an internal node, set to 2), and **min_samples_leaf** (Minimum number of samples required to be at a leaf node, set to 1).

***f) Training Process***

After completing the dataset preprocessing as outlined in ***section (a)*** *above,* the TF-IDF vectorized data was used as input for the Random Forest classifier. The classifier was trained using the optimal hyperparameters identified through the grid search. The model's performance was then evaluated using standard metrics, including accuracy, a classification report, and a confusion matrix, to assess its effectiveness in distinguishing between human-written and machine-generated content.

***2) Model 2: LSTM Neural Network Classifier with GloVe Embeddings (Deep Learning Approach)***

***a) Preprocessing and Vectorization***

In this model, preprocessing plays a critical role in converting raw text data into a format suitable for training the LSTM network. The preprocessing steps include:

**i). Data Augmentation**

Data augmentation refers to the process of creating additional training data by modifying the original dataset in various ways, without changing the underlying label or classification of the data. To enhance the robustness and generalizability of the model, various data augmentation techniques were employed. These techniques included synonym replacement, random deletion, and random swapping of words within sentences. These methods introduce variability into the dataset, helping the model to better generalize across different types of text samples and reduce overfitting (Wei & Zou, 2019).

**ii). Tokenization**

Following data augmentation, the text data was tokenized using a tokenizer that translates words into numerical indices. This step is essential for converting textual data into a format compatible with the LSTM model, as it assigns each word a specific token that the model can understand (Keras, 2024).

**iii). Vectorization with GloVe Embeddings**:

The tokenized text was then converted into dense vector representations using GloVe embeddings. GloVe, which stands for Global Vectors for Word Representation, is a technique that identifies the semantic relationships between words by analyzing global word co-occurrence patterns in a large corpus. In this model, each word was represented as a 100-dimensional vector, providing a detailed representation of the text that maintains the semantic meanings and connections between words (Pennington, Socher, and Manning, 2014).

**iv). Padding**:

Once vectorized, the sequences were padded to achieve consistent sequence lengths across all data samples. This step is crucial for ensuring uniformity during batch processing in the LSTM network, as it guarantees that each sequence has the same length, allowing the model to process the data efficiently and effectively (EITCA Academy, 2023).

***b) Model Implementation***

The LSTM neural network architecture is designed to effectively capture and process the sequential nature of textual data. The key components of the model are as follows:

**i). Embedding Layer**

The model begins with an embedding layer that converts input sequences into 100-dimensional vectors using pre-trained GloVe embeddings. This layer can be set as non-trainable to retain the integrity of the pre-trained embeddings or trainable if fine-tuning is desired. The embedding layer serves as a foundational input that provides the LSTM layers with semantically meaningful representations of words (Brownlee, 2021).

**ii). Bidirectional LSTM Layers**

Following the embedding layer, the model utilized two Bidirectional LSTM layers. The first layer contains 128 hidden units, while the second layer has 64 hidden units. Bidirectional LSTMs are especially useful for text classification because they analyze data sequences in both

forward and backward directions. This bidirectional approach helps the model grasp context from both ends of the sequence, allowing it to better understand the text's context and dependencies (Schuster and Paliwal, 1997; Hochreiter and Schmidhuber, 1997).

#### iii). Dense Layers

After the LSTM layers, the output is fed into a dense layer containing 128 units and utilizing a ReLU activation function. To prevent overfitting, a dropout layer with a 20% dropout rate is applied next, randomly deactivating certain neurons during training. The final layer is a one-dimensional dense layer with a sigmoid activation function, which produces a probability score to determine whether the text is human-written or generated by a machine (Oghaz et al., 2023).

Figure 1 represents the model architecture for the Bidirectional LSTM (Oghaz et al., 2023).

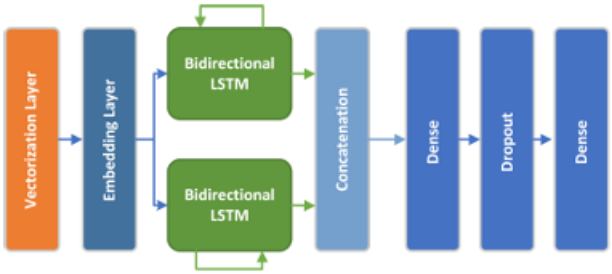


**Fig 1. Model Architecture for Bidirectional LSTM**

### *c) Training Process*

The training process utilized several techniques to optimize model performance and prevent overfitting:

#### i). Optimizer

The Adam optimizer was utilized to adjust the model's weights during training (Kingma and Ba, 2015).

#### ii). Adaptive Learning Rate Scheduling

The '**ReduceLROnPlateau**' scheduler from PyTorch was used for the model implementation in terms of adaptive learning rate scheduling. This technique dynamically adjusted the learning rate during training based on the model's performance on the validation set. Specifically, the learning rate was reduced by a factor of 0.1 if the validation loss did not improve for a specified number of epochs, which in this case was set to a patience of 5 epochs. This approach helped to prevent the model from overshooting the optimal solution and ensured more effective convergence (PyTorch Contributors, 2023).

#### iii). Early Stopping

To prevent the model from overfitting to the training data, early stopping was implemented by closely monitoring the validation loss throughout the training process. The best validation loss was tracked, and an early stopping counter was maintained to count consecutive epochs without improvement. When the validation loss improved, the model's state was saved, and the counter was reset. However, if there was no improvement for 10 consecutive epochs (the specified patience), training was halted. This approach ensured that the model stopped training when it no longer showed improvement on the validation set, thereby optimizing both performance and resource usage (Brownlee, 2019).

### *3) Model 3: BERT-Based Transformer Model (Deep Learning Approach*

### *a) Overview*

BERT (Bidirectional Encoder Representations from Transformers) is a powerful language representation model that utilizes deep bidirectional representations trained on unlabeled text. It employs a masked language model (MLM) and next sentence prediction (NSP) tasks during pre-training, allowing it to capture context from both directions, which is essential for understanding relationships between sentences (Devlin *et al.*, 2019).

For the development of Model 3, the 'BERT-base-uncased' variant (taken from Huggingface) (Hugging Face, 2018) was fine-tuned through techniques like data augmentation and preprocessing, tokenization, data collation & cross validation to classify text as either human-written or AI/ChatGPT-generated.

### *b) Data Preparation*

#### i). Data Augmentation

To improve the model's robustness, data augmentation was applied using the '**nlpaug**' library. Specifically, the 'SynonymAug' augmenter from 'WordNet' was employed to replace certain words in the text with their synonyms. This technique increased the variability within the training data, which is beneficial for enhancing the model's generalizability (Sangani, 2021). Data augmentation was selectively applied to 20% of the dataset to strike a balance between increasing variability and preventing overfitting (Ma, 2019).

#### ii). Data Preprocessing

Following data augmentation, the dataset underwent preprocessing to combine both the original and augmented texts into a comprehensive training set. This process involved cleaning and normalizing the text data, ensuring it was in a format suitable for input into the BERT model. By merging the augmented data with the original dataset, the model was exposed to a richer and more varied training set, which helped it better generalize during the classification task.

### *c) Model Configuration*

#### i). Model Initialization

The BERT-base-uncased model, with 12 transformer layers and 12 attention heads, was initialized using pre-trained weights from the Hugging Face library. Configured for binary classification, the model leveraged BERT's extensive linguistic knowledge and was fine-tuned to distinguish between human-written and machine-generated text (Hugging Face, 2023).

#### ii). Tokenization

Tokenization was an essential preprocessing step conducted before fine-tuning. The BERT tokenizer was employed to transform raw text into tokens that the model

could interpret. This process included padding and truncating text sequences to a maximum of 512 tokens, ensuring consistent input lengths across the dataset. Truncation in this context means shortening or restricting the length of the text sequences. The tokenizer also introduced special tokens, like [CLS] and [SEP], to format the input for BERT. This structured and uniform input enabled the model to efficiently process inputs of varying lengths (Au Yeung, 2020; Devlin et al., 2019; Hugging Face, 2024).

**iii). Data Collation**

A data collator was employed during training to ensure that the tokenized text data was correctly batched. This process involved dynamically padding sequences within a batch to the same length, optimizing memory usage and computational efficiency during the training process (Hugging Face, 2024).

***d) Fine-tuning and Hyperparameter Optimization***

The fine-tuning process involved adjusting the pre-trained BERT parameters and training an additional classification layer on a task-specific dataset. Hyperparameter optimization was a crucial aspect of this fine-tuning, significantly impacting the model's performance. Parameters such as the learning rate, batch size, and the number of training epochs were carefully tuned to enhance the model's efficiency and accuracy. The AdamW optimizer was employed with a learning rate of 1e-5 and a weight decay of 0.05. Training was conducted over 5 epochs with a batch size of 16 for both training and evaluation. This approach allowed the model to retain BERT's deep understanding of language while adapting it to the specific task of classifying text (Hugging Face, 2024).

A cosine learning rate scheduler was used to adjust the learning rate in a cyclical pattern, promoting effective convergence during fine-tuning (Hugging Face, 2024). Additionally, early stopping was implemented to prevent overfitting, halting the training if no improvement was observed in validation loss for five consecutive epochs (Brownlee, 2019).

***4) Model Evaluation***

***a) Common Evaluation Techniques and Metrics***

All three models were evaluated using common evaluation techniques and classification metrics such as data splitting (70/15/15) (Scikit-learn, 2023a), classification reports (Scikit-learn, 2023b), confusion matrices (Scikit-learn, 2023c), accuracy scores (Scikit-learn, 2023d), ROC-AUC scores (Scikit-learn, 2023e), precision-recall curves, and error analysis (MarkovML, 2023; Wikipedia, 2023). These provided a comprehensive assessment of each model's ability to distinguish between human-written and machine-generated text. Additionally, training time was recorded for each model, offering insights into the efficiency of the models during training.

***b) Differences in Evaluation Techniques for Model 3***

The BERT-Based Transformer Model incorporated a unique evaluation technique: a two-fold cross-validation approach (Scikit-learn, 2023f). In this method, the dataset was divided into two subsets, with each subset serving as both a training set and a validation set in different folds. This approach ensured a more robust and generalizable evaluation by thoroughly testing the model across different data splits (MarkovML, 2023).

## IV. Results and Discussion

This section of the project report represents a thorough comparison between the performance of the three models discussed in detail in the Methodology section above. The aim is to identify the optimal model that outperforms the other two in detecting text generated by ChatGPT. The comparison is based on several key metrics, including Accuracy, Confusion Matrices, Receiver Operating Characteristic (ROC) and Precision-Recall Curves, Training Time, and an analysis of Misclassified Examples.

### *A. Accuracy*

The accuracy metric evaluates a model's correctness by determining the proportion of correct predictions out of the total predictions made (MarkovML, 2023). In this project, the training, validation, and testing accuracies of all three models were computed to assess how well each model generalizes to unseen data and to identify any potential issues such as overfitting.

Table 1 below represents the results for the accuracies of the three models.

**Table 1. Training, Validation, and Testing Accuracies**

| Model | Training Accuracy | Validation Accuracy | Testing Accuracy |
|---|---|---|---|
| **Random Forest** | 100.00% | 89.38% | 90.26% |
| **LSTM** | 96.57% | 92.39% | 91.50% |
| **BERT** | 98.69% | 97.75% | 97.50% |

**Analysis:**

The **Random Forest model** achieved a perfect training accuracy of 100.00%, but this dropped significantly to 89.38% on the validation set and 90.26% on the test set. The large gap between training and validation accuracies suggests that the model has likely overfitted to the training data, memorizing patterns instead of learning to generalize well (GeeksforGeeks, 2024).

Secondly, the **LSTM model** showed a strong training accuracy of 96.57%, with validation and testing accuracies of 92.39% and 91.50%, respectively. This smaller gap between training and validation accuracies indicates that the LSTM model managed to generalize better than the Random Forest model, though some overfitting is still evident (GeeksforGeeks, 2024).

Lastly, the **BERT model** demonstrated the highest performance across all phases, with a training accuracy of 98.69%, a validation accuracy of 97.75%, and a test accuracy of 97.50%. The close alignment between these accuracies suggests that BERT effectively avoided overfitting and generalized exceptionally well to new data, making it the most reliable model among the three (GeeksforGeeks, 2024).

### *B. Confusion Matrix*

Confusion matrices offer a comprehensive overview of the model's performance by displaying the counts of true positives, true negatives, false positives, and false negatives. These visual tools provide valuable insights into the classification errors of each model, helping to pinpoint areas where the models excel and where they encounter difficulties (Waterston, 2024).

Figure 2 demonstrates the confusion matrices for the three models of this study.

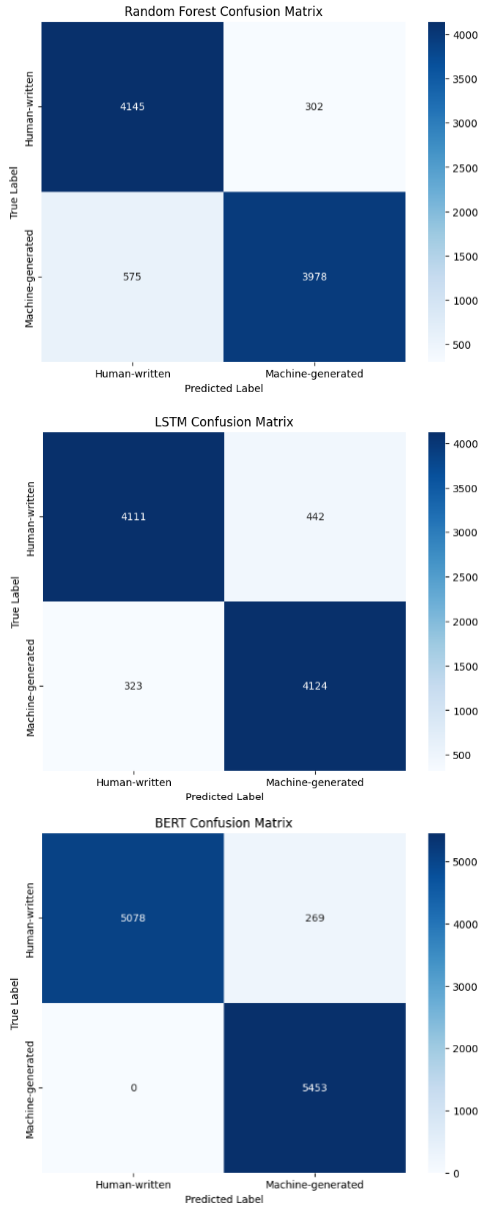


**Fig 2. Confusion matrices for the three models**

The **Random Forest** confusion matrix shows 575 false positives and 302 false negatives, with 4145 true positives and 3978 true negatives. This suggests that the model has significant challenges in accurately distinguishing between human-written and machine-generated text, resulting in a moderate level of misclassification and overall performance. The higher number of false positives indicates a tendency to incorrectly label human-written text as machine-generated, while the false negatives reflect instances where machine-generated text was wrongly classified as human-written.

The **LSTM** model performs better than the Random Forest, with 323 false positives and 442 false negatives. The model correctly identifies 4111 instances of human-written text (true positives) and 4124 instances of machine-generated text (true negatives). While the LSTM model reduces misclassification rates compared to the Random Forest, it still faces difficulties, particularly with false negatives, where it struggles more with correctly identifying machine-generated text as such.

The **BERT** model demonstrates superior performance, with no false positives and 269 false negatives. The confusion matrix indicates 5078 true positives and 5453 true negatives. These results highlight BERT's exceptional ability to accurately classify both human-written and machine-generated text, with minimal errors. This makes BERT the most reliable model in this comparison, as it significantly outperforms the other two models in both precision and recall.

### *C. Receiver Operating Characteristic - Area Under the Curve (ROC-AUC)*

The ROC-AUC value measures a model's ability to distinguish between positive and negative classes, with values closer to 1 indicating better discriminatory performance (Evidently AI Team, 2024).

Figure 3 represents the ROC-AUC curves for the three models.

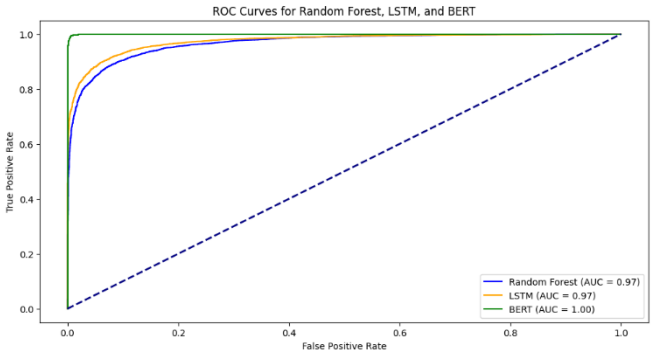


**Fig 3. ROC-AUC Curve for the three models combined**

The ROC-AUC scores for all three models are notably high, indicating strong performance in distinguishing between human-written and machine-generated text. The **BERT model** achieved a perfect ROC-AUC of 1.00, demonstrating exceptional effectiveness. The **LSTM model** also performed very well, with a ROC-AUC of 0.97, matching closely with the **Random Forest model**, which also recorded a ROC-AUC of 0.97. Despite the different accuracies among these models, their ROC-AUC values are closely aligned, suggesting that they all rank predictions well on a probabilistic scale, even if their probability thresholds don't perfectly align (Google Developers, 2024).

### *D. Precision-Recall Curve*

Precision-Recall curves show the trade-off between precision and recall at different thresholds, highlighting a

model's ability to identify positive instances accurately, especially in imbalanced datasets (Bonnet, 2023).

Figure 4 below demonstrates the precision-recall curve for the three models.

The Precision-Recall curve indicates that the **BERT model** significantly outperforms the other two models, maintaining very high precision across almost all levels of recall. This demonstrates BERT's strong capability in distinguishing between human-written and machine-generated text, with minimal trade-offs between precision and recall. The **LSTM model** performs slightly less effectively than BERT, showing a gradual decline in precision as recall increases, indicating that it is slightly less capable of maintaining high precision when capturing all relevant instances. The **Random Forest model** performs the least effectively, exhibiting a more pronounced drop in precision as recall increases, suggesting that it struggles more to maintain precision when aiming for higher recall. Overall, BERT is the best performer, followed by LSTM, with Random Forest trailing behind.

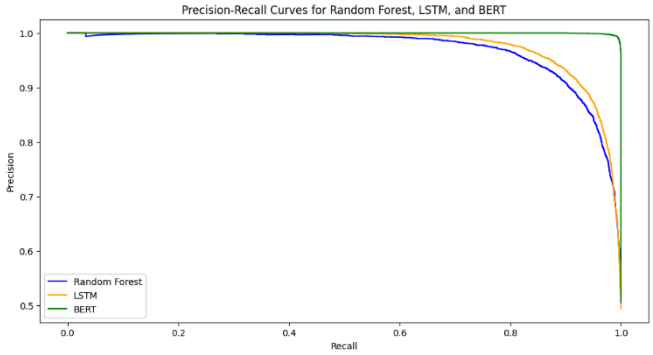


**Fig 4. Precision-Recall Curves for the three models**

### *E. Training Time*

Training time refers to the duration required to train each model and produce predictions on the test data. This metric provides an understanding of each model's computational efficiency, a crucial consideration when evaluating a model for practical deployment (Alamleh et al., 2023).

**TABLE 2. TRAINING TIME FOR EACH MODEL**

| Model | Training Time |
|---|---|
| **Random Forest** | 1822.69 seconds |
| **LSTM** | 33.99 seconds |
| **BERT** | 6916.45 seconds |

Table 2 represents the training time for all three models, highlighting significant differences in computational efficiency. All the models were trained and tested using the A100 high performance GPU. The **BERT model** took the longest time, at 6916.45 seconds, due to its complex architecture and intensive computation needs. The **LSTM model**, with a simpler structure, completed training much faster, in just 33.99 seconds. The **Random Forest model**, taking 1822.69 seconds, was faster than BERT but slower than LSTM. This longer training time for Random Forest compared to LSTM is primarily due to the ensemble nature of Random Forest, which involves training multiple decision trees independently and aggregating their results (sylvaticus, 2020; Khanna, 2024).

### *F. Misclassified Example Analysis*

Misclassified Example Analysis examines instances where a model incorrectly predicts labels, highlighting the types of errors the model tends to make. For example, in our case, the Random Forest model misclassified the text, "**Fruits are a vital component of a healthy diet, offering a rich source of essential vitamins, minerals, and antioxidants**." It predicted the text as human-written when it was actually machine-generated, likely due to the text's straightforward style, which closely mimics human writing.

## V. CONCLUSION

Among the models tested, the BERT-based transformer emerged as the most effective, achieving the highest test accuracy (97.50%) and ROC-AUC score (1.00). Despite requiring the most computational time (6916.45 seconds), its superior performance in identifying AI-generated text was evident. The LSTM model also performed commendably, with a test accuracy of 91.50% and a ROC-AUC score of 0.97. It offered a significantly faster training time of 33.99 seconds, making it a more computationally efficient option. The Random Forest model, while less accurate (90.26% accuracy, 0.97 ROC-AUC), provided valuable insights into the trade-offs between simpler machine learning approaches and more advanced deep learning techniques with a model training time of 1822.69 seconds. Overall, the study emphasizes the urgent need for continued enhancements in AI detection techniques to maintain the integrity of digital communications as AI-generated content becomes increasingly prevalent.

## VI. FUTURE WORK

In extending the current work, future research should focus on expanding datasets, including complex examples like the Human-ChatGPT Polished Paired Texts (HPPT) dataset, to enhance the detection of subtle AI modifications. Additionally, broadening the dataset to cover various domains, such as legal, medical, and creative writing, will help assess the models' robustness across different contexts. Moreover, developing new detection models or fine-tuning existing ones through adversarial training will be essential to counter the growing sophistication of AI-generated content. Incorporating classical machine learning models like SVM and KNN, alongside deep learning models like RoBERTa, will provide a more comprehensive comparison. Lastly, addressing ethical considerations, such as minimizing detection biases, will ensure these systems remain fair and reliable in real-world applications.

## ACKNOWLEDGMENTS

I am profoundly grateful to my project supervisor, Professor Arkaitz Zubiaga, for his invaluable guidance and support throughout this project. His expertise was crucial in shaping my research. I also extend my gratitude to the faculty and staff of the School of Electronic Engineering and Computer Science (EECS) at QMUL for their helpful feedback and assistance, which significantly contributed to my academic journey.

## REFERENCES


[1] Aggarwal, K., Mijwil, M.M., Sonia, Al-Mistarehi, A-H., Alomari, S., Gök, M., Zein Alaabdin, A.M., and Abdulrhman, S.H., 2022. Has the Future Started? The Current Growth of Artificial Intelligence, Machine Learning, and Deep Learning. *Iraqi Journal for Computer Science and Mathematics*, 3(1), pp.115-123. Available at: https://doi.org/10.52866/ijcsm.2022.01.01.013 [Accessed 11 Aug. 2024].

[2] Oghaz, M.M.D., Dhame, K., Singaram, G., and Saheer, L.B. (2023). Detection and Classification of ChatGPT Generated Contents Using Deep Transformer Models. *IEEE Access*, pp. 1-14. DOI: 10.1109/ACCESS.2023.0322000.

[3] Cingillioglu, I. (2023). Detecting AI-generated essays: the ChatGPT challenge. *The International Journal of Information and Learning Technology*, 40(3), pp. 259-268. DOI: 10.1108/IJILT-03-2023-0043.

[4] Flanagin, A., Bibbins-Domingo, K., Berkwits, M. and Christiansen, S.L., 2023. Nonhuman "Authors" and Implications for the Integrity of Scientific Publication and Medical Knowledge. *JAMA*, 329(8), pp.637-639. doi:10.1001/jama.2023.1344.

[5] Grimaldi, G. and Ehrler, B., 2023. AI et al.: Machines are about to change scientific publishing forever. *ACS Energy Letters*, 8, pp.878-880. Available at: https://doi.org/10.1021/acsenergylett.2c02828 [Accessed 11 August 2024].

[6] Azamfirei, R., Kudchadkar, S.R. and Fackler, J., 2023. Large language models and the perils of their hallucinations. *Critical Care*, 27(1), p.75. https://doi.org/10.1186/s13054-023-04393-x.

[7] Mindner, L., Schlippe, T., and Schaaff, K., 2023. Classification of Human- and AI-Generated Texts: Investigating Features for ChatGPT. *arXiv*. https://arxiv.org/abs/2308.05341v1 [Accessed 11 August 2024].

[8] Bhat, A., 2023. *GPT-wiki-intro* [dataset]. Available at: https://huggingface.co/datasets/aadityaubhat/GPT-wiki-intro [Accessed 11 Aug. 2024]. DOI: 10.57967/hf/0326.

[9] Alamleh, H., AlQahtani, A. A. S., & ElSaid, A. (2023). Distinguishing Human-Written and ChatGPT-Generated Text Using Machine Learning. *Proceedings of the 2023 Systems and Information Engineering Design Symposium (SIEDS)*.

[10] Desaire, H., Chua, A.E., Isom, M., Jarosova, R. and Hua, D., 2023. Distinguishing academic science writing from humans or ChatGPT with over 99% accuracy using off-the-shelf machine learning tools. *Cell Reports Physical Science*, 4, p.101426. https://doi.org/10.1016/j.xcrp.2023.101426.

[11] Fröhling, L. and Zubiaga, A., 2021. Feature-based detection of automated language models: tackling GPT-2, GPT-3 and Grover. PeerJ Computer Science, 7. Available at: https://doi.org/10.7717/peerj-cs.443 [Accessed 13 August 2024].

[12] Liang, W., Yuksekgonul, M., Mao, Y., Wu, E. & Zou, J., 2023. GPT detectors are biased against non-native English writers. *arXiv*. Available at: https://doi.org/10.48550/arXiv.2304.02819.

[13] Yang, L., Jiang, F. and Li, H., 2023. Is ChatGPT Involved in Texts? Measure the Polish Ratio to Detect ChatGPT-Generated Text. *arXiv:2307.11380v2 [cs.CL]*.

[14] Mitchell, E., Lee, Y., Khazatsky, A., Manning, C.D. and Finn, C., 2023. *Zero-Shot Machine-Generated Text Detection using Probability Curvature*. Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, pp.1-13.

[15] Bommasani, R. et al., 2022. *On the Opportunities and Risks of Foundation Models*. Stanford University. Available at: https://arxiv.org/abs/2108.07258 [Accessed 13 Aug. 2024].

[16] Rashidi, H.H., Fennell, B.D., Albahra, S., Hu, B. and Gorbett, T., 2023. The ChatGPT conundrum: Human-generated scientific manuscripts misidentified as AI creations by AI text detection tool. *Journal of Pathology Informatics*, 14, p.100342. Available at: http://dx.doi.org/10.1016/j.jpi.2023.100342 [Accessed 13 August 2024].

[17] Dergaa, I., Chamari, K., Zmijewski, P. and Ben Saad, H., 2023. From human writing to artificial intelligence generated text: examining the prospects and potential threats of ChatGPT in academic writing. *Biology of Sport*, 40(2), pp.615-622. Available at: https://doi.org/10.5114/biolsport.2023.125623.

[18] GeeksforGeeks. (2024) *Text Preprocessing in NLP*. Available at: https://www.geeksforgeeks.org/text-preprocessing-in-nlp/ (Accessed: 17 August 2024).

[19] Karabiber, F., 2024. *TF-IDF — Term Frequency-Inverse Document Frequency*. LearnDataSci. Available at: https://www.learndatasci.com/glossary/tf-idf-term-frequency-inverse-document-frequency/ [Accessed 17 August 2024].

[20] Scikit-learn developers. (2024) *sklearn.feature_extraction.text.TfidfVectorizer*. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html (Accessed: 17 August 2024).

[21] BotPenguin. (2024) *Stop Words*. Available at: https://botpenguin.com/glossary/stop-words (Accessed: 17 August 2024).

[22] Breiman, L. (2001) 'Random forests', *Machine Learning*, 45(1), pp. 5-32. Available at: https://doi.org/10.1023/A:1010933404324 [Accessed 17 August 2024].

[23] Scikit-learn developers. (2024) *sklearn.ensemble.RandomForestClassifier*. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html (Accessed: 17 August 2024).

[24] Selvaraj, N. (2022) *Hyperparameter Tuning Using Grid Search and Random Search in Python*. Available at: https://www.kdnuggets.com/2022/10/hyperparameter-tuning-grid-search-random-search-python.html (Accessed: 17 August 2024).

[25] Koehrsen, W. (2018) 'Hyperparameter Tuning the Random Forest in Python', *Towards Data Science*, 10 January. Available at: https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74 (Accessed: 17 August 2024).

[26] Wei, J. & Zou, K., 2019. EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks. *arXiv preprint*, arXiv:1901.11196v2.

[27] Keras. (2024) *Tokenizer base class*. Available at: https://keras.io/api/keras_nlp/tokenizers/tokenizer/ (Accessed: 18 August 2024).

[28] Pennington, J., Socher, R., and Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, October 25-29, pp. 1532-1543.

[29] EITCA Academy, 2023. *Why is it necessary to pad sequences in natural language processing models?* [online] Available at: https://eitca.org/artificial-intelligence/eitc-ai-tff-tensorflow-fundamentals/natural-language-processing-with-tensorflow/training-a-model-to-recognize-sentiment-in-text/examination-review-training-a-model-to-recognize-sentiment-in-text/why-is-it-necessary-to-pad-sequences-in-natural-language-processing-models/ [Accessed 18 August 2024].

[30] Brownlee, J., 2021. *How to Use Word Embedding Layers for Deep Learning with Keras*. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/ [Accessed 18 August 2024].

[31] Schuster, M. and Paliwal, K.K., 1997. Bidirectional Recurrent Neural Networks. *IEEE Transactions on Signal Processing*, 45(11), pp.2673-2681.

[32] Hochreiter, S. and Schmidhuber, J., 1997. Long Short-Term Memory. *Neural Computation*, 9(8), pp.1735-1780.

[33] Kingma, D.P. and Ba, J.L., 2015. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*. Available at: https://arxiv.org/abs/1412.6980 [Accessed 18 Aug. 2024].

[34] PyTorch Contributors. (2023) *torch.optim.lr_scheduler.ReduceLROnPlateau*. Available at: https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.ReduceLROnPlateau.html (Accessed: 18 August 2024).

[35] Brownlee, J., 2019. *A Gentle Introduction to Early Stopping to Avoid Overtraining Neural Networks*. [online] MachineLearningMastery.com. Available at: https://machinelearningmastery.com/early-stopping-to-avoid-overtraining-neural-network-models/ [Accessed 18 August 2024].

[36] Hugging Face. (2018). *BERT base model (uncased)*. Available at: https://huggingface.co/google-bert/bert-base-uncased (Accessed: 18 August 2024).

[37] Devlin, J., Chang, M-W., Lee, K. and Toutanova, K., 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805v2*.

[38] Sangani, R. (2021) *Powerful text augmentation using NLPAUG*. Towards Data Science. Available at: https://towardsdatascience.com/powerful-text-augmentation-using-nlpaug-5851099b4e97 (Accessed: 18 August 2024).

[39] Ma, E. (2019) *nlpaug: Synonym Augmenter*. Available at: https://nlpaug.readthedocs.io/en/latest/augmenter/word/synonym.html (Accessed: 18 August 2024).

[40] Hugging Face. (2023) *BertForSequenceClassification*. Available at: https://huggingface.co/transformers/v3.5.1/model_doc/bert.html#bertforsequenceclassification (Accessed: 18 August 2024).

[41] Lenovo. (2024) *What is truncate?* Available at: https://www.lenovo.com/gb/en/glossary/what-is-truncate/?orgRef=https%253A%252F%252Fwww.perplexity.ai%252F (Accessed: 18 August 2024).

[42] Au Yeung, A., 2020. *Mastering BERT Tokenization and Encoding*. [online] Available at: https://huggingface.co/transformers/index.html [Accessed 18 August 2024].

[43] Hugging Face, 2024. *Padding and truncation - Transformers documentation*. [online] Available at: https://huggingface.co/docs/transformers/padding_and_truncation [Accessed 18 August 2024].

[44] Hugging Face. (2024) *Data collator*. Available at: https://huggingface.co/docs/transformers/en/main_classes/data_collator (Accessed: 19 August 2024).

[45] Hugging Face. (2024) *A Full Training*. Available at: https://huggingface.co/learn/nlp-course/chapter3/4?fw=pt (Accessed: 19 August 2024).

[46] Scikit-learn (2023a) *Cross-validation: evaluating estimator performance*. Available at: https://scikit-learn.org/stable/modules/cross_validation.html (Accessed: 19 August 2024).

[47] Scikit-learn (2023b) *Classification Report*. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html (Accessed: 19 August 2024).

[48] Scikit-learn (2023c) *Confusion Matrix*. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html (Accessed: 19 August 2024).

[49] Scikit-learn (2023d) *Accuracy Score*. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html (Accessed: 19 August 2024).

[50] Scikit-learn (2023e) *ROC-AUC Score*. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html (Accessed: 19 August 2024).

[51] Scikit-learn (2023f) *Cross-Validation: K-Folds*. Available at: https://scikit-learn.org/stable/modules/cross_validation.html#k-fold (Accessed: 19 August 2024).

[52] Wikipedia (2023) *Error Analysis*. Available at: https://en.wikipedia.org/wiki/Error_analysis (Accessed: 19 August 2024).

[53] MarkovML (2023) *Model Evaluation Metrics: Methods & Approaches*. Available at: https://www.markovml.com/blog/model-evaluation-metrics (Accessed: 19 August 2024).

[54] GeeksforGeeks, 2024. *Validation vs. Test vs. Training Accuracy. Which One is Compared for Claiming Overfit?* [online] Available at: https://www.geeksforgeeks.org/validation-vs-test-vs-training-accuracy-which-one-is-compared-for-claiming-overfit/ [Accessed 20 Aug. 2024].

[55] Waterston, S. (2024) *An Introduction to Understanding Confusion Matrix in Machine Learning*. Available at: https://www.pickl.ai/blog/an-introduction-to-understanding-confusion-matrix-in-machine-learning/ (Accessed: 20 August 2024).

[56] sylvaticus (2020) *My Random Forest is very slow*. Available at: https://discourse.julialang.org/t/my-random-forest-is-very-slow/45471 (Accessed: 20 August 2024).

[57] Khanna, V. (2024) *Random Forests in Machine Learning for Advanced Decision-Making*. Available at: https://shelf.io/blog/random-forests-in-machine-learning/ (Accessed: 20 August 2024).

[58] Evidently AI Team, 2024. *How to explain the ROC curve and ROC AUC score?* [online] Evidently AI. Available at: https://www.evidentlyai.com/classification-metrics/explain-roc-curve [Accessed 20 August 2024].

[59] Google Developers. (2024). *Classification: ROC and AUC*. [online] Available at: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc [Accessed 20 Aug. 2024].

[60] Bonnet, A., 2023. *Accuracy vs. Precision vs. Recall in Machine Learning: What is the Difference?* Encord Blog, 23 November. Available at: https://encord.com/blog/classification-metrics-accuracy-precision-recall/ [Accessed 20 August 2024].

## VII. Appendix

The appendix section of the report includes additional figures and tables. Table 3 represents the schema of the **aadityaubhat/GPT-wiki-intro**' dataset which includes the detailed description of the structure and contents of the dataset.

**Table 3. Schema of the GPT-Wiki-Intro Dataset (Bhat, 2023)**

| Column | Datatype | Description |
|---|---|---|
| **id** | int64 | ID |
| **url** | string | Wikipedia URL |
| **title** | string | Title |
| **wiki_intro** | string | Introduction paragraph from Wikipedia |
| **generated_intro** | string | Introduction generated by GPT (Curie) model |
| **title_len** | int64 | Number of words in title |
| **wiki_intro_len** | int64 | Number of words in wiki_intro |
| **generated_intro_len** | int64 | Number of words in generated_intro |
| **prompt** | string | Prompt used to generate intro |
| **generated_text** | string | Text continued after the prompt |
| **prompt_tokens** | int64 | Number of tokens in the prompt |
| **generated_text_tokens** | int64 | Number of tokens in generated text |

Figure 5 demonstrates a random sample of the dataset after it has been labelled, balanced and sampled as explained in the Data Preprocessing section of the Methodology above.

| | text | label |
|---|---|---|
| **0** | Anam Cara is a phrase that refers to the Celti... | 0 |
| **1** | is a Japanese light novel series written by Ni... | 0 |
| **2** | The OA vz. 27 (Obrněný automobil vzor 27) was ... | 0 |
| **3** | The Last Hurrah is a 1956 novel written by Edw... | 0 |
| **4** | Sherman, Clay & Co. was an American musical in... | 0 |
| **...** | ... | ... |
| **59995** | Two fossil crania were discovered along the ba... | 1 |
| **59996** | William Whitehead (born 23 March 1970) is an E... | 1 |
| **59997** | Mirijevo is an urban neighborhood of Belgrade... | 1 |
| **59998** | Sea Moon (foaled 6 March 2008) is a British Th... | 1 |
| **59999** | The 2013 Summer Tour was a co-headlining conce... | 1 |

**Fig 5. Dataset after labelling and balancing**

Table 4 represents the classification report results for the three models, covering both human and machine-generated text. A classification report is a summary of the model's performance. It includes key metrics like:

- **Precision**: How accurate the model is at identifying AI-generated text.
- **Recall**: How well the model identifies all instances of AI-generated text.
- **F1 Score**: A balance between precision and recall.
- **Support**: The number of actual cases for each class (AI-generated or human-written).

**TABLE 1. CLASSIFICATION REPORT TABLE FOR THE THREE MODELS FOR BOTH HUMAN & MACHINE-GENERATED TEXT**

| Model | Precision (Human) | Recall (Human) |
|---|---|---|
| **Random Forest** | 0.88 | 0.93 |
| **LSTM** | 0.93 | 0.90 |
| **BERT** | 1.00 | 0.95 |

| Model | F1-Score (Human) | Support (Human) |
|---|---|---|
| **Random Forest** | 0.90 | 4447 |
| **LSTM** | 0.91 | 4553 |
| **BERT** | 0.97 | 5347 |

| Model | Precision (Machine) | Recall (Machine) |
|---|---|---|
| **Random Forest** | 0.93 | 0.87 |
| **LSTM** | 0.90 | 0.93 |
| **BERT** | 0.95 | 1.00 |

| Model | F1-Score (Machine) | Support (Machine) |
|---|---|---|
| **Random Forest** | 0.90 | 4553 |
| **LSTM** | 0.92 | 4447 |
| **BERT** | 0.98 | 5453 |