International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 02 | Feb - 2025 SJIF Rating: 8.448 ISSN: 2582-3930

(LLM) AI Generated Text Detection

Prof Sayali Shivarkar[1], Kunal Pisolkar[2], Hitesh Pawar[3], Bhaskar Prajapati[4], Nishant Nimbalkar[5]

[1]Professor, Dept of Computer Engineering, JSPM's Jayawantrao Sawant College of Engineering Pune

[2,3,4,5]Student, Dept of Computer Engineering, JSPM's Jayawantrao Sawant College of Engineering Pune

Abstract:

Large Language Models have completely revolutionized text generation, with significant studies reporting an unprecedented quantity of human-quality content. Still, this super capability comes with some grave risks: the spreading of misinformation at an unprecedented scale, academic plagiarism, and even erosion of trust in written communication. We are therefore developing a robust AI-generated text detection system. Our two-phased approach first involves training a neural network classifier on at first carefully hand-crafted textual features designed to capture subtle variations between human and LLM-generated text, then creating a web application using Next.js that allows a user to easily input text to analyze and receive a clear classification outcome. This project is yet another contribution to combat the spread of misinformation and to the preservation of academic integrity. But most importantly, for the responsible and ethical use of powerful AI technologies.

Keywords: *AI-Generated Text, Text Detection, Large Language Models, Machine Learning, Academic Integrity, Misinformation, Authenticity*

I. INTRODUCTION

Large Language Models like Chat GPT, Claude, and Gemini have taken the world by storm. Incredible is this power to generate human-quality text in a matter of seconds. Applications abound, from chatbots to writing assistants—a new paradigm in how we interact with language has come about. But this transformational power surely is a sword that cuts both ways. The same technology that can write a poem or summarize a scientific article can also concoct convincing fake news, plagiarize academic work, and spread misinformation with alarming speed. In this way, the paper raises urgent questions about how to ensure responsible AI use and trusted written communication. Our project will address this challenge head-on: we will develop a reliable system that detects AI-generated text.

Motivation:

These fully-capable LLMs-artifact producing high-quality texts, just like humans, demand coherent detection methods. The motivation for the research is due to the fact that AI-generated text misuse is gradually growing and may turn into the spread of misinformation, academic dishonesty, and attacking the trustful circle of online content. Coming up with reliable detection techniques is important for maintaining authenticity and integrity in today's world that is driven by artificial intelligence-generated information.

II. LITERATURE SURVEY

Debora Weber-Wulff et al. "Testing of detection tools for AI-generated text" [1] with the arrival of generative models such as ChatGPT, which generate output that is practically indistinguishable from that produced by students. Consequently, AI detection in text is becoming an imperative within academic integrity. Recently, they reviewed a range of free and commercial AI detection tools; the capability of Turnitin and PlagiarismCheck in distinguishing AI from human-generated output was brought forth. Some of the main limitations this tool was found to have in the research were its extreme bias toward labeling texts as human-written, and also its very limited accuracy in cases of AI content that undergo machine translation or paraphrasing, as most of those bypassed detection. In general, while these tools provide a decent starting point to detect AI-generated text, their inconsistencies underpin the need for further technological innovation to provide integrity that is clearly defined within an AI-driven educational environment.

Eric Mitchell et al. "DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature" [2]. The unprecedented growth in large language models, like GPT-3 and Chat-GPT, made AI-generated text spread into every aspect. That very factor brings up urgent needs for effective mechanisms of its detection. However, machine-generated text is hard to detect because of the fluency of generated texts by these models. Many studies

testified that people cannot distinguish between AI- and human-generated content. Therefore, the traditional methods of detection using supervised approaches suffer from the drawbacks of overfitting to a domain or model and are hence poor in generalizing capability Uchendu et al. (2020)[7]. Zero-shot methods like DetectGPT use model-specific probability structures to identify the text. In case of DetectGPT, for an instance, a curvature-based approach has been considered with hypothesis that AI-generated contents fall into a negative curvature area in model's log probability function enabling its detection without any extra training data Mitchell et al. (2023)[2]. Many zero-shot baseline methods have been outperformed with this method, and it provides a very promising approach toward scalable AI-generated text detection on diverse domains.

Elizabeth Clark et al. "All That's 'Human' Is Not Gold: Evaluating Human Evaluation of Generated Text" [3], the authors investigate the degree to which untrained human judges are able to discern AI-generated text from that written by humans; overall, judges did little better than chance, with the worst performance associated with text from more advanced models like GPT-3. In nearly every case, this research found, evaluators tended to make judgments about text based on superficial aspects of text—that is, grammar or style—rather than coherence of content or factual accuracy, which led to poor judgments. Poor success was achieved in attempts to further improve the evaluators precision
with brief training sessions, using examples and guidance on detecting machine patterns. This study therefore underlines the limitation of using untrained human judgment as a benchmark in evaluating the quality of machine-generated text.

Köbis and Mossink "Artificial intelligence versus Maya Angelou: Experimental evidence that people cannot differentiate AI-generated from human- written poetry" [4] examined whether one can reliably distinguish AI-generated poetry from human-generated, finding in most cases that this indeed is difficult to do by participants, at least when the best AI outputs have been selected by humans. Their experiment used GPT-2 to generate poems, paired with human-written poems, and showed that in many such cases participants could not successfully classify AI text. This suggests that contemporary NLG models are capable of very persuasive creative text. What's more, participants' preferences for human poems did not shift even after being told that the author of a given poem was a machine—a fact highly relevant to the challenges of algorithmic detection in creative domains.

Vinu Sankar Sadasivan et al. "Can AI-Generated Text be Reliably Detected?" [5] investigate the robustness of AI-text detectors, the reliability of which, even for state-of-the-art tools, remains a severe problem. Based on several varieties of recursive paraphrasing attacks that can easily evade previously suggested detection methods using neural networks, zero-shot, and watermarking, the authors conducted their work in this area. The results of their work showed remarkable weaknesses in the reliability of such detectors. Indeed, it was found in this work that recursive paraphrasing impacts only a bit the quality of text, but significantly deteriorates the performance of detectors, whose AUROC scores drop by more than 50%. The most important finding of this research is the fact that it points to an urgent need for finer AI-text detection techniques as paraphrasing attacks are increasingly sophisticated.

III. METHODOLOGY

A) **Sequential Classifier**

1. **TF-IDF Feature Extraction**
   a) **Input**: A dataset (dataframe df1) containing data in 'text' column and corresponding labels in 'label' column indicating whether the text is human or AI generated.
   b) **Vectorization**: The text data was transformed into representations vocabulary size to the top 5000 most frequent words.
   c) **Output**: A dataFrame where each row represents a text sample, and each column represents a word from the vocabulary, with cell values being the TF-IDF scores.
2. **Data Splitting**
   a) **Input**: TF-IDF feature vectors and corresponding labels.
   b) **Splitting**: The data is split into training and testing sets (80/20 split) using train_test_split.
   c) **Output**: Training data as X_train, y_train and testing data (X_test, y_test).

3. **Model Building**
   a) **Model Architecture:** A Sequential Classifier consisting of
      i. A dense input layer with 128 units and ReLU activation.
      ii. A dropout layer with a rate of 0.3.
      iii. Another dense layer with 64 units and ReLU activation.
      iv. Another dropout layer with a rate of 0.3.
      v. An output layer with a single unit and sigmoid activation.
   b) **Compilation:** The model is compiled using the Adam optimizer (learning rate=0.001), binary cross-entropy loss, and accuracy metric
4. **Model Training**: The model is trained using the fit method for 15 epochs with a batch size of 32. Training and validation data are used (X train,y train encoded, X test, y test encoded). Labels are assumed to be encoded (eg 0 and 1).
5. **Model Evaluation**: The model's performance is evaluated on the test data using the evaluate method. Test loss and test accuracy are reported.

B) **BERT Model**

1. **Preprocessing and Tokenization**
   a) **Input**: A text dataset with the text in a column named 'text' and the sentiment labels in another column called 'label.'
   b) **Tokenization**: Text data will be tokenized using the BertTokenizer from the transformers library. It utilizes the 'bert-base-uncased' pre-trained model, whereby all text is lowercased by the model. Padding and truncation have been used to make the sequences all of similar length - max_length=128. In this way, the tokenizer returns input_ids and attention masks.
   c) **Output**: X_input_ids, tokenized input sequences; X_attention_masks, masks showing valid tokens.
2. **Data Split**
   a) **Input**: Tokenized input IDs, X_input_ids; labels, y; attention masks, X_attention_masks.
   b) **Splitting**: This function will split the data into training and testing sets using an 80/20 split via the train_test_split function.
   c) **Output**: X_train, y_train, X_train_attention for training; X_test, y_test, X_test_attention for testing.
3. **Class Weight Calculation**
   a) **Input**: Training labels (y_train)
   b) **Weight Calculation**: Class weights are calculated using the compute class weight functionality from *sklearn.utils.class* weight with the '*balanced*' strategy to take any class imbalance of the training data into account.
   c) **Output**: a dictionary class weights dict that maps class labels to their respective weights.
4. **Model Loading & Compilation**
   a) **Model Initialization**: Load the *TFBertForSequenceClassification* pre-trained model, *'bert-base-uncased'*, and set it for two labels.
   b) **Optimizer**: An *AdamW* optimizer is instantiated with the create optimizer from transformers with a learning rate of $2e^{-5}$ using a linear learning rate schedule without any warm-up steps. The number of the training steps is defined from the size of the training data, *batch* size, and the number of epochs. Batch size here is 16 and the number of *epochs* is 5.
   c) **Compilation of the Model**: The model was compiled using AdamW optimizer and *SparseCategoricalCrossentropy* loss-which

works with integer labels-and accuracy as the metric for evaluation.

5. **Data Preparation for Training**
   a) **Creation of Dataset**: The dataset would utilize tf.data.Dataset.from tensor slices for creating the objects of training and validation sets. Input features include input IDs and attention masks, while labels correspond to targets. Then, the data is batched according to the batch size specified.
6. **Model Training**: By using the factual function, it would fit the model with the instance training dataset given, the validation dataset, number of epochs, and class weights. Training progress is given verbosely.
7. **Model Evaluation**: Once trained, the model evaluates the test set via the evaluate method. It reports test loss and test accuracy.

IV. RESULTS

A) Sequential Classifier

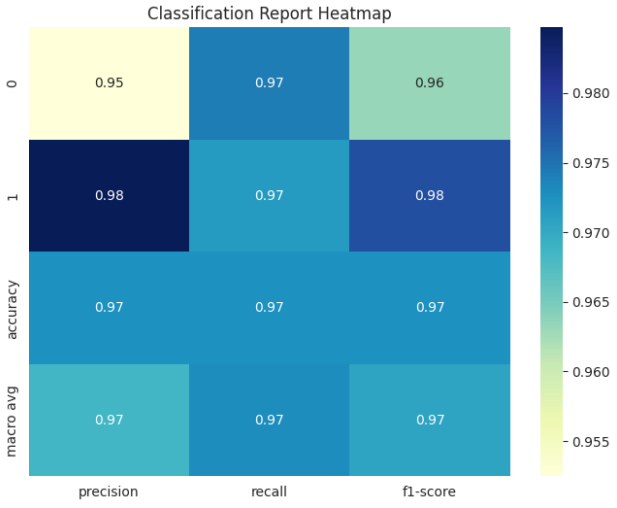


Figure 4.1: Classification Report Sequential

This heatmap represents the classification report, which allows a thorough performance analysis of a binary classification model. Rows come with different aspects of the report: class-specific metrics for 0 and 1, and overall metrics: accuracy and macro average. Columns are for precision, recall, and F1-score—key evaluation metrics in classification. The color intensity reflects the score for each metric, with darker blue meaning higher values. Class 1 has very high precision, recall, and F1-score—all being 0.98 or 0.97—which means the classifier is really performing great identifying that class. Class 0 has a slightly low precision of 0.95 but is also very good in recall and F1-score, obtaining 0.97 and 0.96, respectively. With a general accuracy of about 0.97, both the overall and macro average are really high, meaning that generally, the model performs well.

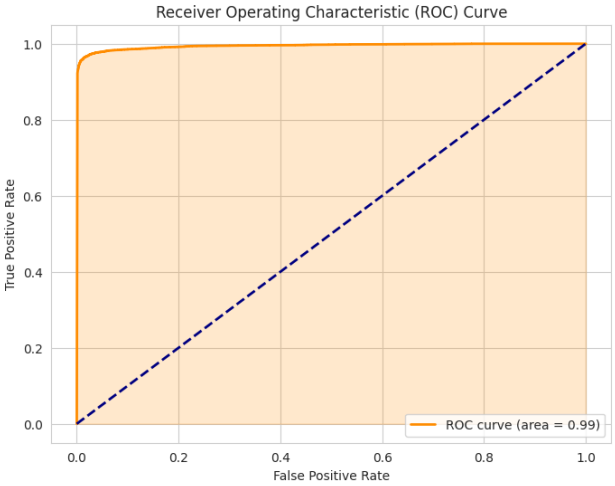


Figure 4.2 : AUC ROC Curve Sequential

The above plot depicts a ROC curve, which is a common way to summarize the performance of a binary classifier. The orange curve depicts the ROC curve itself—that is a plot of the TPR versus FPR across different classification thresholds. The dashed navy blue line depicts the performance of a classifier choosing at random. The performance of AUC under the ROC curve is reported to be 0.99, which is very close to the perfect value of 1. This very high AUC represents great discriminatory power and effectively suggests that the model performs very well in distinguishing between the considered classes—for example, AI-generated text versus human-written text.

The orange shading encapsulates the difference in performance between this model and a random classifier. Its proximity to the top-left corner is further confirmation of the strong performance of the model.

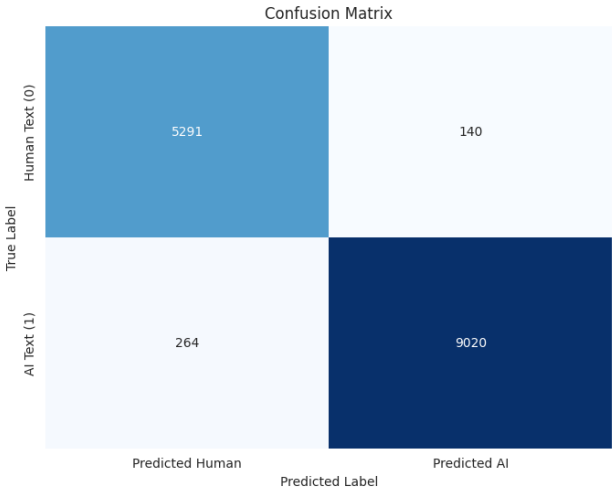


Figure 4.3 : Confusion Matrix Sequential

The plot shown here is a confusion matrix, often used for visualizing how well a classification model is performing – in this case likely distinguishing between AI-generated text and human-written text. From the raw human-written texts (True Label: Human Text), the model correctly predicted 5291 as human-written text (Predicted Human) but classified 140 as AI-generated text (Predicted AI) as shown in the matrix. For true AI text samples (True Label: AI Text), 264 were falsely classified as human (Predicted Human), and 9020 were accurately predicted to be from an AI source (Predicted AI). The darker a square is shaded, the more instances belong to that class. This suggests the model is functioning well overall but has a slightly higher error rate for human-written text classification than AI-generated.

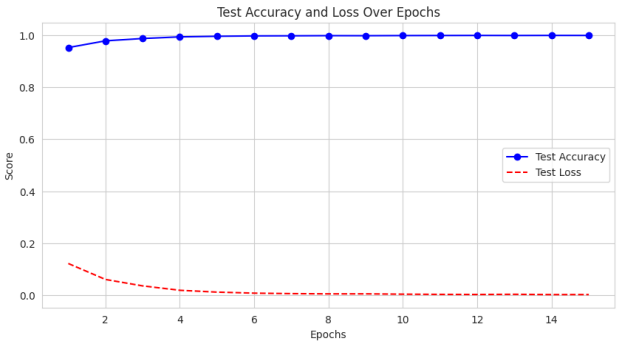
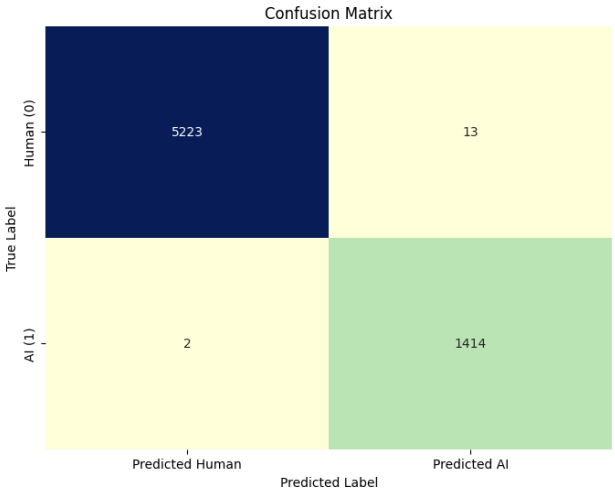


Figure 4.4 :Training Curve Sequential

The graphical representation shows the performance of a machine learning model through 15 epochs. The blue line, which represents test accuracy, increases very quickly from about 0.95 at epoch 1 to nearly 1.0 by epoch 3 and then seems to plateau and remain at a persistently high value for the remainder of the training period. This suggests that the model learns the patterns underlying the test data very rapidly. On the other hand, the red dashed line, representing test loss, decreases significantly from around 0.13 at epoch 1 and nearly reaches zero by epoch 5 and stays there afterward. This loss decrease patterns are in line with the upward trend in accuracy, demonstrating the model's improved ability to classify the test data accurately with fewer mistakes at each subsequent epoch. The fast balance between accuracy and loss proves that the model is adapted well to the dataset and the applied training method.

B) **BERT Model**



The confusion matrix above shows that a BERT model, apparently in classification tasks, is effective. Detecting text as human-written or AI-generated. The matrix shows that of the 5223 original texts created from humans (Ground Truth: Human (0)), the model accurately classified 5223 as written by humans (Predicted Human) and misclassified only 13 as AI-generated (Predicted AI). Similarly, 1414 texts of actual AI-generation (True Label: AI (1)), it correctly classifies 1414 as AI-generated (Predicted AI with) only 2 misclassified as if written by humans. The very low numbers of misclassifications, highlighted by the lightness of color for the off-diagonal

Figure 4.5 : Confusion Matrix BERT

squares, and the high. The number of actual classifications-as indicated by the darker colors on the diagonal squares Results reveal that the BERT model is to perform well with this task.

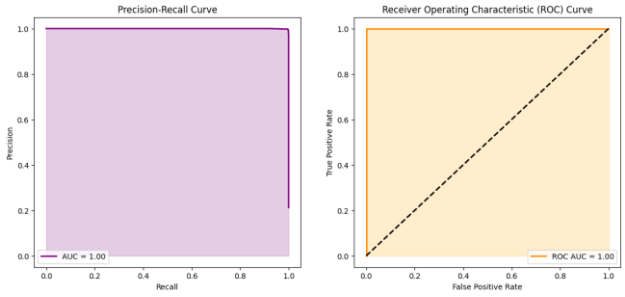


Figure 4.6 : PR and AUC ROC Curve

The graph shows two curves widely used in the assessment of a binary classification model. The left plot illustrates the Precision-Recall curve. In the Precision-Recall curve the precision is plotted against the recall. The purple line is perfect performance, always obtaining a precision of 1.0 at all values of recall, and the shaded purple area beneath is an AUC of 1.0. This means that the model is able to obtain perfect recall without losing precision. The right plot displays the Receiver Operating Characteristic curve, plotting the True Positive Rate against the False Positive Rate. That orange line on the right plot is the top-left boundary, so that's a perfect classification. The black dashed line plots the random classifier, and orange shaded area corresponds to an ROC AUC of 1.0, which ensures that the model has an extraordinary capability in classification between the classes. The curves depict nearly perfect performance of the model for the given dataset.

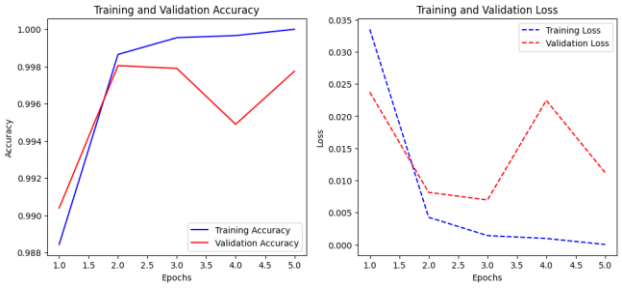


Figure 4.7 : Training Curve BERT

This plot gives two curves showing the training and validation performance of the error curve over five epochs by the machine learning BERT model discussed above. The plot on the left is accuracy; the blue line is the training accuracy; The red line represents validation accuracy. They both start high and rise rapidly, though the validation accuracy hovers and fluctuates around the second epoch. While accuracy with training lags to increase. The right plot shows loss, with the dashed blue line showing training loss and the dashed red line showing validation loss. The train and validation loss drops steeply at first but then has a strong spike up to the fourth epoch, meaning maybe overfitting, while the training loss keeps decreasing in a smooth curve. The gap is between the continuously improving training metrics and the fluctuating validations metrics further reinforce the possibility of overfitting, indicating the model might be learning the training data too well at the expense of generalizing to unseen data.

V. CONCLUSION

AI-generated text presents various challenges and opportunities in most sectors. So far, the project has researched the detection capability and limit of existing methods against AI-generated content with an emphasis on recognizing and combating misinformation, as well as protecting academic integrity and creative works. We used a base sequential model and one based on BERT to improve the efficiency and robustness of AI text detection. Findings from this project suggest innovation in technique for detection, adaptation into changing their language model, and ethical practice in AI development. Through an advanced technological change, proactive steps through AI text detection will be of fundamental importance in engendering the needed trust within the

digital environment and having equity assessments within education.

VI. ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude to our guide Prof. Sayali Shivarkar for their able guidance and support. We would also like to extend our gratitude to the Principal Mr. R. D. Kanphade for providing us with all the facilities that was required. We would also like to acknowledge with much appreciation the role of our staff. We sincerely thank our parents and our friends who have been always helping and encouraging us in every situation.

REFERENCES

1. Weber-Wulff, D., Anohina-Naumeca, A., Bjelobaba, S., Foltýnek, T., Guerrero-Dib,J., Popoola, O., Šigut, P., & Waddington, L. (2023). Testing of Detection Tools for AI-Generated Text. *International Journal for Educational Integrity*, 19(1), 26.

2. Mitchell, E., Lee, Y., Khazatsky, A., Manning, C. D., & Finn, C. (2023). Detect-GPT: Zero-Shot Machine-Generated Text Detection Using Probability Curvature. In *Proceedings of the 40th International Conference on Machine Learning*, 24841–24863.

3. Clark, E., August, T., Serrano, S., Haduong, N., Gururangan, S., & Smith, N. A. (2021). All That's 'Human' Is Not Gold: Evaluating Human Evaluation of Generated Text. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, 2789–2802.

4. Kobis, N. C., & Mossink, L. D. (2021). Artificial Intelligence Versus Maya Angelou: Experimental Evidence that People Cannot Differentiate AI-Generated from Human- Written Poetry. *Computers in Human Behavior*, 114, 106553.

5. Sadasivan, V. S., Kumar, A., Balasubramanian, S., Wang, W., & Feizi, S. (2024). Can AI-Generated Text be Reliably Detected? *arXiv preprint arXiv*:2303.11156.

6. Gehrmann, S., Strobelt, H., & Rush, A. (2019). GLTR: Statistical Detection and Visualization of Generated Text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 111–116, Florence,Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-3019.

7. Uchendu, A., Le, T., Shu, K., & Lee, D. (2020). Authorship Attribution for Neural Text Generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 8384–8395, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.673.