DOI: 10.6977/IJoSI.202506_9(3).0005
H. Panigrahi, S. Naidu, etc./Int. J. Systematic Innovation, 9(3), 75-95 (2025)

Handwriting match and AI content detection

Hrishikesh Panigrahi, Siddhanth Naidu, Ambuj Pandey, Phiroj Shaikh*, Amiya Kumar Tripathy

Department of Computer Engineering, Don Bosco Institute of Technology, Mumbai, India

*Corresponding author E-mail: phiroj@dbit.in

(Received 22 November 2024; Final version received 12 February 2025; Accepted 09 April 2025)

Abstract

Machine-generated text presents a potential threat not only to the public sphere but also to education, where the authenticity of genuine students is compromised by the presence of convincing, synthetic text. There are also concerns about the spread of academic misconduct, particularly direct replication among students. In response to these challenges, this paper introduces the Handwriting Match and Artificial Intelligence (AI) Content Detection System (HMAC). HMAC utilizes optical character recognition (OCR) mechanisms to convert handwritten and typed content from a single-page portable document format into machine-readable text, thus enabling further analysis. Drawing on recent advances in natural language understanding, HMAC aims to preserve the educational value of assignments by effectively detecting AI-generated content. In addition, HMAC has a strong plagiarism detection system that uses a comparative analysis of student submissions in a particular academic field. This paper describes HMAC's architecture, methodology, and results, emphasizing its key contributions: improved handwritten content extraction with OCR and improved identification of AI-generated content. The study addresses the research question of how HMAC improves the identification of AI-generated content and supports academic integrity compared to other methodologies.

Keywords: Academic Assessment, Artificial Intelligence Content Detection, Document Analysis, Similarity Detection, Transformer-Based Models

1. Introduction

A new era of convenience and accessibility has been brought about by the widespread deployment of generative models, as demonstrated by ChatGPT, which has completely changed the landscape of academic assignments in recent years. However, there have been challenges along the way with this evolution, including a concerning trend of students using artificial intelligence (AI)-generated content directly in their assignments because of the ease of accessing such tools in obtaining the content required for the assignments. When students choose easily accessible generative content above the learning process, assignments lose their inherent value, which is a danger to the educational process.

In response to this critical issue, our research introduces the Handwriting Match and AI Content Detection System (HMAC) as a robust and multifaceted solution. HMAC is meticulously designed to recognize and address the inappropriate incorporation of AI-generated content in student assignments, particularly those submitted in handwritten or typed form, often in Portable Document Format (PDF) Romero et al. (2012). By leveraging transformer-based models fine-tuned on the GPT-wiki-intro dataset, HMAC employs state-of-the-art technology to discern the nuanced differences between human-generated and AI-generated content.

As AI systems become increasingly integrated into our daily lives, understanding the factors influencing their acceptance and interaction becomes paramount. Pelau et al. (2021) explored the intricate dynamics of human-AI interaction, examining the roles of interaction quality, empathy, and perceived anthropomorphic characteristics in shaping the acceptance of AI within the service industry. Daniel et al. (2019) delved into the delicate balance between AI and human behavior, emphasizing the need to comprehend and prevent potential harm that may arise from AI systems acting like humans. Uzun (2023) investigated academic integrity concerns related to ChatGPT, shedding light on methodologies

for detecting AI-generated content in educational settings. In addition, Sadasivan et al. (2023) probed the reliability of detecting AI-generated text, questioning the effectiveness of existing methodologies.

In the field of language understanding, the advent of Bidirectional Encoder Representations from Transformers (BERT) marked a significant paradigm shift. Kenton and Toutanova (2019) introduced BERT as a pre-training model for deep bidirectional transformers, showcasing its prowess in language understanding. These endeavors collectively constitute the backdrop against which we explore the evolving landscape of AI and its intricate intersections with human dynamics.

Beyond content identification, HMAC's primary objective is to retain and impart the educational value that comes with academic assignments. HMAC positions generative models as supplemental tools, encouraging students to participate in the learning process before incorporating AI-generated content, as opposed to using it as a quick fix. This paradigm change fosters a deeper comprehension of the subject matter, which is essential for the holistic development of students.

At its core, HMAC employs an optical character recognition (OCR) mechanism to convert both handwritten and typed content into machine-readable text. This process involves several steps, including word detection through platforms such as Roboflow, an ordering algorithm to arrange words into coherent lines, and integration with OCR models such as Microsoft's transformer-based OCR (trocr)-base-handwritten, for precise word recognition and sentence formation. The system simultaneously applies an AI content recognition model to the OCR-processed text, revealing information about the proportions of AI and human content. The second aspect of our investigation delves into the world of OCR, a technology critical to deciphering and extracting information from visual data. Wu et al. (1997) laid a foundational groundwork for text extraction from images, a critical step in the OCR process. Subsequently, Li and Doermann (1998) pioneered automatic text tracking in digital videos, propelling OCR capabilities into dynamic visual contexts. Kim (1999) introduced local color quantization for automatic text location in complex color images, advancing OCR techniques in handling intricate visual scenarios. Jain and Yu (1998) further expanded OCR capabilities into images and video frames, contributing to the development of comprehensive OCR methodologies.

One of HMAC's unique features is its integration of a plagiarism check, which involves comparing uploaded assignments to a database of assignments related to the same subject. By taking a comprehensive approach, instances of academic dishonesty among students are reported, and AI-generated content is identified. The system generates detailed reports for users, including percentages of AI-generated content and plagiarism detection results.

HMAC provides a complete and proactive solution that essentially guards against the improper application of generative models in educational settings. Assignments are meant to be instructive. This platform provides educators with the resources they need to keep a strict evaluation environment in place. Examining the design, process, and outcomes of HMAC, this work offers a viable answer to the changing problems associated with academic integrity in the digital era of research.

In this work, we address the following research question: (i) How does HMAC significantly improve the detection of AI-generated work, especially in terms of academic integrity, compared to current approaches? (ii) How does the OCR aspect of our technology enhance the process of extracting and interpreting handwritten text from photographs, especially for assignments and academic documents?

2. Related Work

Several recent studies have focused on detecting and analyzing AI-generated content, particularly in terms of academic integrity and the responsible use of AI. Rodriguez et al. (2022) investigated the cross-domain detection of GPT-2-generated technical text, shedding light on the challenges and implications of identifying machine-generated content across different domains. Mitrović et al. (2023) explored decision explanations for machine learning models, specifically in the context of distinguishing between ChatGPT and human-generated text, emphasizing the importance of interpretability in AI-based detection systems.

Addressing academic integrity concerns, Uzun (2023) conducted an investigation into ChatGPT's impact on academic settings, proposing methods for detecting AI-generated content to preserve academic integrity. Mindner et al. (2023) delved into the classification of human- and AI-generated texts, exploring features for effectively identifying content generated by models such as ChatGPT.

Wahle et al. (2023) introduced the concept of AI Usage Cards, aiming to facilitate the responsible reporting of AI-generated content by providing transparent and informative details about the AI models involved. In the realm of evasion strategies, Lu et al. (2024) discussed the challenges of large language models being guided to evade AI-generated text detection, highlighting the need for robust detection mechanisms Jauhiainen et al. (2016), Lindén et al. (2012).

Several studies have made significant contributions to the field of OCR, with each focusing on a different aspect of accuracy, efficiency, and

post-correction methodologies Lund et al (2013), Lund et al (2011), Reynaert (2010).

Dong and Smith (2018) proposed a multi-input attention mechanism for unsupervised OCR correction, demonstrating its effectiveness in enhancing OCR accuracy Wick et al. (2018). The similar type work has been done by Evershed and Fitch (2014), Hämäläinen and Hengchen (2019), Kauppinen (2016), Koistinen et al. (2017), Kettunen et al. (2018), Llobet et al. (2010), Silfverberg et al. (2016), Springmann et al. (2014), Vobl et al. (2014). Guha et al. (2019) presented Devnet, an efficient convolutional neural network architecture for handwritten Devanagari character recognition, contributing to the advancement of OCR techniques for specific script recognition. The effective use of the neural networks in such areas has been done by Graves et al. (2006) and Srivastava (2014).

Kettunen and Koistinen (2019) explored the utilization of the open-source OCR engine Tesseract for re-OCR of Finnish Fraktur, providing insights into quality improvement strategies. Li et al. (2021) introduced TrOCR, a transformer-based OCR model with pre-trained models, showcasing the potential of transformer architectures in OCR tasks.

Sabu and Das (2018) conference paper, "*A Survey on Various Optical Character Recognition Techniques*," was presented at the Conference on Emerging Devices and Smart Systems 2018 in India. This manuscript sheds light on various OCR methods and provides useful insights into their applications. It also recommends future research avenues into advanced OCR techniques and their potential applications in document analysis.

3. Related Work

3.1. Design

The HMAC proposed requires users to input their assignments in PDF format through a specific web interface. The system then uses an OCR process to convert typed or written text into machine-readable text. To provide users with visibility into both human and AI content percentages, the system simultaneously runs the OCR-processed text through an AI content detection model to ascertain the likelihood that the material was generated by AI. By comparing the submitted work with a database of assignments in the same subject, HMAC also includes a plagiarism check feature. The system then generates a thorough report for users, including the percentage of material generated by AI and the findings of any plagiarism detection.

3.2. Architecture

A single-page PDF document is first fed into the process and transformed into machine-readable text. To find duplicate material and establish whether the content is AI generated, this text is processed and put through match detection and AI content detection. The processed file is kept in a database together with its metadata and analysis findings. The system verifies that each file has been examined and compared. If not, a different file is retrieved from the database for processing. The final comparison findings and AI content detection results are displayed after all files have been analyzed.

The architecture depicted in Fig. 1 ensures a comprehensive and systematic approach to content analysis in PDF files. It seamlessly integrates PDF-to-text conversion, match detection, AI content detection, and database management, offering a robust solution for identifying duplicate and AI-generated content while maintaining the integrity of the analyzed material.

3.3. Method of Data Collection

A thorough literature search was conducted to identify relevant research articles on AI-generated content detection techniques. Databases such as IEEE Xplore, ACM Digital Library, arXiv, and Google Scholar were searched using relevant keywords, such as "Generative Models Analysis," "AI Content Detection," and "Optical Character Recognition (OCR)." Following our literature search, we carried out a two-step quality assessment process. In the first step, we screened the identified studies for relevance to the scope of our study. Manuscripts that did not directly address the topic or were unrelated to our HMAC project were not further considered.

Then, we rigorously evaluated their scientific quality. This evaluation included an assessment of each manuscript's methodology, including the research design, data collection techniques, and experimental setup. The articles were chosen based on their relevance to the topic and contribution to the existing body of knowledge in the field. Only peer-reviewed articles and conference papers were considered for the review. Duplicate articles, non-English articles, and articles irrelevant to the scope of HMAC were excluded.

4. OCR in Handwriting Match and AI Content Detection System

OCR is a critical stage in the complex framework of HMAC. OCR is the cornerstone in our effort to convert complex handwritten characters into text that can be read by a computer when integrating handwritten assignments into our system. Handwritten text has intricacies that must be extracted, processed, and understood. This intricate process is broken down into multiple steps, each with a distinct objective.

The first step is converting handwritten assignments in PDF format into image files. This conversion creates the foundation for further OCR processes using the pdf2image library. Then, using Roboflow, the system explores word detection, exposing the minute nuances of each word in the handwritten document. The challenges posed by variations in handwriting styles and irregular word sequencing are addressed using an innovative algorithm called the Word Sequencing Algorithm, ensuring accurate and ordered detection. Once the words are identified and ordered, the system uses OCR models, with a focus on Microsoft's trocr-base handwritten model. This specialized OCR model excels at recognizing and transcribing each word in handwritten content, bridging the analog and digital dimensions of education. The stages are outlined below and explained in detail.

The flowchart in Fig. 2 depicts the step-by-step process used in the OCR system.

4.1. Document Image Generation

The first step of the proposed software is to process a single-page PDF input in real-time. After uploading the PDF, the backend code quickly converts it to a JPG image using the pdf2image Python library. This library requires an external utility called Poppler to render PDFs. It is worth noting that while Poppler is commonly found on Linux systems, it may require additional steps to install on Windows. After saving the image, the system seamlessly proceeds to the next stage: semantic segmentation with Roboflow.

4.2. Semantic Segmentation with Roboflow

Following the successful conversion of the PDF to an image, the Word Detection stage focuses on precisely detecting each word within the converted image. For this task, we used Roboflow. Roboflow

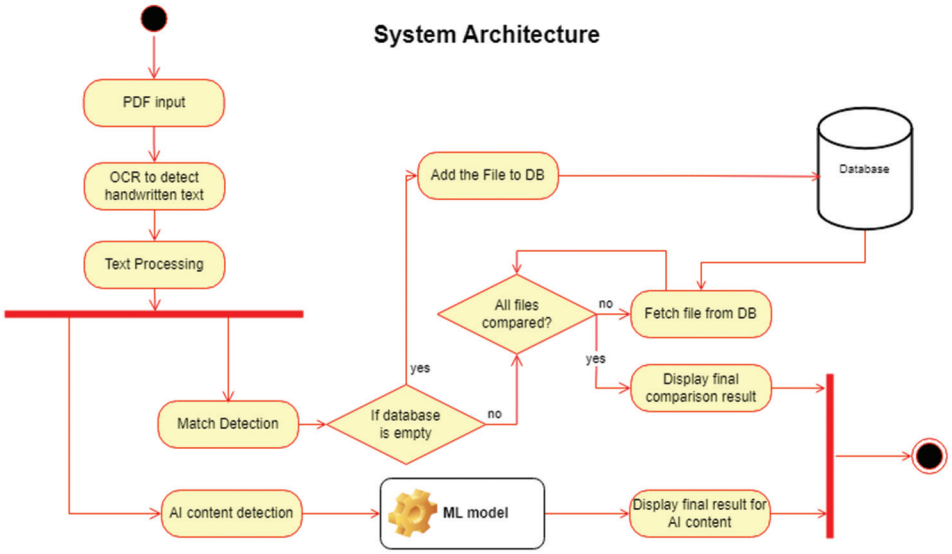


Fig. 1. Handwriting Match and Artificial Intelligence Content Detection System Architecture
Abbreviations: AI: Artificial intelligence; DB: Database; ML: Machine learning; OCR: Optical character recognition; PDF: Portable document format.

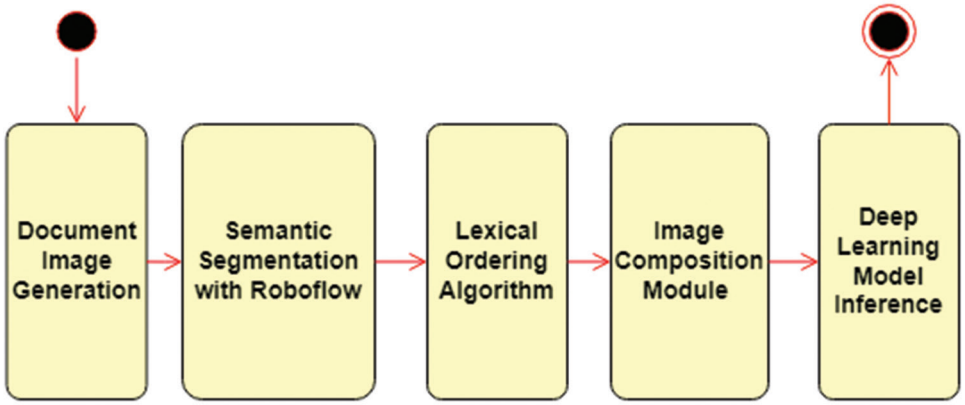


Fig. 2. Overview of the optical character recognition workflow

is the preferred option due to its tailored capabilities for detecting words in handwritten documents. The platform's robust capabilities are used to identify each word in the handwritten document.

The Roboflow model's results are stored in an array called "box_dimension," with each element representing the properties of a detected word: "*x*" for the *x* coordinate, "*y*" for the *y* coordinate, "width" for the word's width, and "height" for its height. Simultaneously, another array is created to hold the images for each detected word. These images are created using the data stored in the "box_dimension" array, with each property facilitating the extraction of a specific portion of the image. Each extracted word image is then saved to a designated folder.

For instance, the "box_dimensions" array looks like this.

[{'index': 1, 'x': 602, 'y': 360, 'width': 102, 'height': 39},
{ 'index': 2, 'x': 33, 'y': 144, 'width': 86, 'height': 35},
{ 'index': 3, 'x': 691, 'y': 706, 'width': 79, 'height': 34},
{ 'index': 4, 'x': 404, 'y': 613, 'width': 100, 'height': 36},
{ 'index': 5, 'x': 214, 'y': 707, 'width': 89, 'height': 28}]

However, a problem arises due to the potential disorder in the detection order, resulting in a mismatch between the sequence of words in the original PDF and the order in which they are detected. Although it might seem reasonable to sort the "box_dimension" array first by *x* values and then by *y* values, the second sorting step overrides the first. As a result, a more sophisticated algorithm is required to ensure proper word sequencing while preserving the original content's integrity.

4.3. Lexical Ordering Algorithm

The ordering algorithm 1 is critical for organizing the detected words and reconstructing the original flow of sentences and paragraphs from an unordered set of word images. This algorithm takes a systematic approach to grouping words into lines of text based on their vertical *y* positions in the document and then further groups them based on their horizontal *x* positions.

This algorithm is intended to arrange words into lines according to their spatial coordinates, specifically their vertical *y* and horizontal *x* positions within a given "box_dimensions" list. Initially, the list of word bounding boxes is sorted by *y* coordinates to group words on the same line. The algorithm iterates through each word, keeping track of which words appear on the same horizontal line. This is maintained by the current_line list. The algorithm uses a threshold value (20 in this case) to determine if a word belongs to the current line by comparing the *y* distance of the current word with the previous word in the current_line. If the difference in the y distance falls within the threshold, the word is added to the current_line; otherwise, the current_line is sorted by *x* coordinates (left-to-right order), appended to the lines list, and reset to start a new line with the current word. This process continues until all words are processed. Finally, the final current_line is sorted to the *x* coordinates and added to the lines list to ensure the algorithm correctly sequences words into lines as they appear in the text layout.

This algorithm is crucial for structuring the text, as it efficiently groups words into lines based on their vertical positions. It is critical for establishing the correct word sequence within each line, resulting in a structured representation of the original content.

4.4. Image Composition Module

The ordering algorithm is critical for organizing the detected words and reconstructing the original flow of sentences and paragraphs from an unordered set of word images. This algorithm takes a systematic approach to grouping words into lines of text based on their vertical *y* positions in the document and then groups them based on their horizontal *x* positions.

In the following stage, using the correct word sequence obtained in the previous step, an OCR model was used to determine the textual content of

Algorithm 1: Lexical Ordering

```
function WordSequencing (box_dimensions)
    box_dimensions ← Sorted (box_dimensions,
    key=lambda box: (box["y"]))
    lines ← []
    current_line ← [box_dimensions[0]]
    threshold← 20

     for i in range (1, len (box_dimensions)) do
         if box_dimensions[i]["y"]−current_line[−1]["y"]
         < threshold then
            ▷ Add the word to the current line if it is on
            the same line
                Append (current_line, box_dimensions[i])
            else ▷ Sort the current line by x-value and add
            it to the lines list
                 current_line← Sorted (current_line,
    key=lambda box: (box["x"]))
                Extend (lines, current line)
                current_line ← [box_dimensions[i]] ▷
                Reset the current line to the current word
            end if
        end for

    ▷ Sort the last line by x-value and add it to the
lines list
     current_line←Sorted (current_line, key=lambda
     box: (box["x"]))
    Extend (lines, current_line)
    return lines
end function
```

each detected word. The traditional method involves processing each word individually through the OCR model, resulting in corresponding text outputs. However, this method can be time-consuming, particularly in cases where the document contains a large number of words.

To address this issue, the proposed software uses a strategic technique known as image concatenation. Image concatenation is the process of combining two or more images to create a single composite image. Rather than processing each word individually, the software combines 10 images into a single line using concatenation. This concatenated image is then fed into the OCR model for analysis. By concatenating multiple images into a single line, the software not only speeds up the OCR process but also increases overall efficiency. This technique is especially useful when dealing with a large volume of text, resulting in a more streamlined and resource-efficient workflow.

One notable challenge in implementing the image concatenation technique is the potential lack of discernible padding between two concatenated images. OCR models may find it challenging to recognize and distinguish individual words inside the combined image due to the absence of spacing. A calculated approach is used to solve this problem: putting white space between two concatenated photos. White padding improves the accuracy of OCR. This ensures that the OCR process can effectively interpret each word in the concatenated line, allowing these words to be seamlessly combined into a coherent sentence.

4.5. Deep Learning Model Inference

Following the concatenation of lines, each composite line is processed using Microsoft's trocr-base handwritten model. This OCR model is designed specifically for the recognition of handwritten text, using advanced deep-learning techniques to ensure precise identification and transcription of each word in the handwritten content. The model's emphasis on handwritten text recognition makes it an appropriate choice for accurately interpreting the complexities of handwritten assignments.

4.5.1. Model architecture

The TrOCR model is architecturally designed around the transformer framework, which includes both an image transformer and a text transformer. The dual-transformer architecture depicted in Fig. 3 is fundamental to TrOCR's ability to accurately extract visual features from images and perform language modeling for OCR (Li et al., 2021).

The transformer architecture is implemented in a standard encoder-decoder configuration within TrOCR. The encoder component is specifically engineered to capture representations of image patches, leveraging the visual information inherent in the input. The decoder, on the other hand, is responsible for generating a workpiece sequence, guided not only by the visual features extracted from the image but also by the predictions made in the preceding steps. TrOCR utilizes the conventional transformer encoder-decoder structure, which emphasizes its adaptability and effectiveness in dealing with both image-based and language-related tasks. This architectural choice highlights the transformer model's versatility, allowing it to seamlessly integrate image processing and language generation components within a unified framework. The above diagram presents the architectural design of the model.

- *Encoder:* The encoder part processes the input image and extracts high-level features that represent the content of the image. In the context of OCR, this could involve understanding the shapes and patterns of characters.
- *Decoder*: The decoder part interprets the features generated by the encoder and produces the final output, which is the recognized text. The decoder considers the context of characters and their relationships to improve accuracy. The decoder uses self-attention masking to prevent gaining more information during training than prediction. The attention mask ensures that the output for position *i* only considers the previous output and input for positions less than *i* (Li et al., 2021).

$$h_i = Proj(Emb(Token_i)) \tag{1}$$

$$\sigma\left(h_{ij}\right) = \frac{e^{h_{ij}}}{\sum_{k=1}^{v}\left(e^{h_{ik}}\right)} \qquad for\ j = 1, 2, ..., V \tag{2}$$

A linear layer projects the decoder's hidden states from the model dimension to the dimension of the vocabulary size *V*, and the softmax function calculates the probabilities over that vocabulary. We used beam search to obtain the final result.

5. AI Text Detection

5.1. Dataset

Rapid advancements in large language models such as GPT have brought enormous potential and unexpected challenges. One such challenge is the widespread use of GPT-generated text, which raises questions about its authenticity and potential for misuse. To meet this critical need, a dataset was used specifically for detecting text generated by GPT (Bhat, 2023). Table 1 provides a detailed breakdown of the dataset's columns, which include identifiers, Wikipedia URLs, titles, Wikipedia introduction paragraphs, and corresponding content generated by the GPT (Curie). It

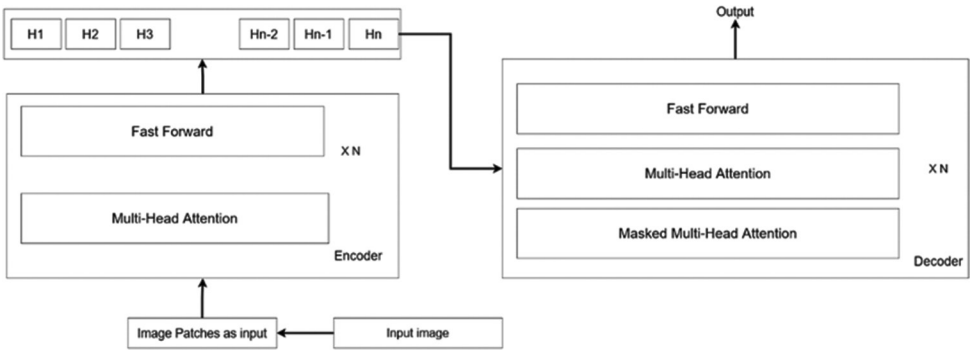


Fig. 3. Architecture of the transformer-based optical character recognition model

Table 1. Summary of dataset columns and their descriptions

Column	Data type	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

also includes length metrics and token counts for titles, Wikipedia introductions, generated introductions, prompts, and the text that follows the prompt, providing a comprehensive overview of the dataset structure.

5.1.1. Feature engineering

To optimize model performance and focus on the most salient features, we applied a strategic feature engineering process. This involved carefully selecting the following essential features:

(i) ID: A unique identifier assigned to each text introduction, enabling efficient data management and tracking

(ii) Text: The raw text content of the introduction, serves as the primary input for the GPT-detection model

(iii) Label: A binary classification label, explicitly indicating whether the text was generated by a human (0) or GPT (1). This label serves as the ground truth for model training and evaluation.

After modification, the dataset structure has been simplified for specific analysis. The "ID" column now displays the identifier in string format, while the "text" column combines data from both "wiki_ intro" and "generated_intro." The "Label" column, which is important for classification, is introduced, with 0 representing human-generated content and 1 representing AI-generated content. Table 2 displays the results of feature engineering applied to the dataset, which resulted in a consolidated "Text" column by combining content from "wiki_intro" and "generated_ intro." The "Label" column represents the classification label, with "0" indicating human-generated content and "1" indicating AI-generated content. These engineered features form the foundation for training and evaluating classification models that distinguish between human-generated and AI-generated text.

5.2. Model

The selection of an appropriate model was critical for the AI text detection process. Transformer-based models, particularly DistilBERT (Bidirectional Encoder Representations from Transformers), well-known for their ability to understand natural language, were evaluated and chosen for their adaptability to the task at hand. The chosen model was then meticulously trained using the annotated dataset. The dataset was divided into training, validation, and test sets, and the model's performance was optimized based on continuous evaluation and validation results (Sanh et al., 2019).

To determine whether the input text was created by AI or by humans, the AI text detection model generates a probabilistic evaluation. The possibility

that the text was produced by AI is represented by one percentage, whereas the likelihood that it was authored by humans is represented by another.

5.2.1. Model architecture

The architecture of the DistilBERT model, a condensed form of BERT, for AI text identification, is displayed in Fig. 4. It begins with an input layer that handles text sequences up to a certain maximum length (512 tokens, for example). The pre-trained DistilBERT model receives these text sequences as input after they have been transformed into token embedding.

The pre-trained model (DistilBERT) acts as the core component, which uses its bidirectional attention mechanism to comprehend contextual relationships in the input text. DistilBERT is effective for tasks such as text classification as it minimizes the number of parameters while maintaining the majority of BERT's representational capacity. After passing through the pre-trained model, the text representations are routed to a fine-tuned layer. This layer represents the specific modifications and task-specific training that were applied to the pre-trained model to help it adapt to the AI text detection task. The fine-tuned model's output is passed through a linear layer pre-classifier, reducing dimensionality and preparing the data for further processing. The representations are fed into a fully connected layer, which maps the features to the final classification task. The final stage is the topic tagging layer, which does the classification. It generates probabilistic scores that indicate whether the input text is human-generated (class 0) or AI-generated (class 1). These probabilities provide interpretable information about the text's likely source.

Table 2. Engineered features for classification

Column	Datatype	Description
ID	int64	ID
Text	string	Text taken from wiki_intro & generated_intro
Label	int64	0 for human, 1 for AI

In short, during the forward pass, the input IDs and attention mask are processed by the pre-trained DistilBERT layer, resulting in a series of hidden states. These hidden states are sent through a fine-tuned layer, a linear pre-classifier, and fully connected layers. The final output layer creates a two-dimensional vector that represents the probabilities for both human-generated and AI-generated text.

6. Similarity

A parallelized comparison methodology has been used in the test of uploaded assignments for similarity, a critical component of the HMAC. By utilizing concurrent futures and a ThreadPoolExecutor, it effectively compares the contents of an uploaded file with several entries in the database, saving the findings for additional examination. A ThreadPoolExecutor was used to start the parallel processing mechanism. ThreadPoolExecutor in Python facilitates concurrent task execution via threads, which is ideal for independent tasks such as file similarity comparisons. It optimizes resource use by reusing threads and scales tasks efficiently by adjusting thread pool size. Asynchronous execution enables tasks to run independently, enhancing responsiveness. Compared to multiprocessing, which uses separate processes with higher memory overhead, and manual threading, which requires more complex thread management, ThreadPoolExecutor offers a simpler, more efficient solution. It submits tasks in parallel to compare the content of the uploaded file with each file model in the list. The comparisons list stores the outcomes for later review and documentation.

6.1. Stages for Detecting Similarity

For similarity detection, first, HMAC calculates the TF-IDF for the documents and then uses cosine similarity to compare the number of similar terms present in the currently uploaded document to the documents present in the database.

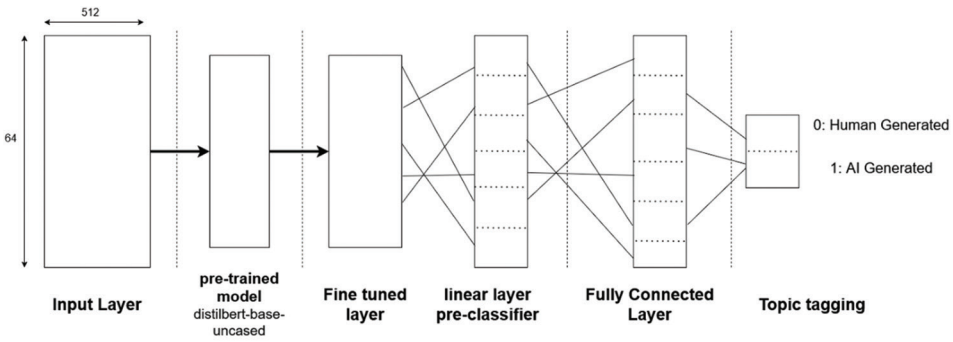


Fig. 4. Architecture of the artificial intelligence (AI) text detection model

6.1.1. Term Frequency-Inverse Document Frequency (TF-IDF)

TF-IDF is a statistical measure that evaluates the importance of a term in a document relative to its occurrence in a collection of documents. The TF-IDF score for a term t in document d is calculated as follows:

TF−IDF(t,d,D)

$$TF\text{–}IDF(t, d, D) = TF(t, d) \times IDF(t, D) \quad (3)$$

$$IDF(t,d) = \log\left(\frac{Total\ no.of\ documents\ in\ corpus\ d}{No.of\ documents\ with\ term\ t + 1}\right) +$$

$$1TF(t,d) = \frac{Number\ of\ terms\ t\ appears\ in\ document\ d}{Number\ of\ terms\ in\ document\ d} \quad (4)$$

where:

(i) TF (t, d) is the term frequency of term t in document d

(ii) IDF (t, D) is the inverse document frequency of term t in the document corpus d.

In HMAC, the TF-IDF matrix is utilized to represent the importance of each term in the uploaded assignment and other submissions, forming the basis for similarity calculations. Below is an excerpt from the output using a sample set of documents:

(0, 129) 0.1857772922586975
(0, 106) 0.1857772922586975
(0, 99) 0.1857772922586975
(0, 127) 0.1857772922586975
(0, 53) 0.1857772922586975
(0, 119) 0.1857772922586975
(0, 43) 0.1857772922586975
(0, 3) 0.1857772922586975…

In the example above, the line (0, 129) 0.1857772922586975 indicates that in the first document, the TF-IDF score for the term represented by the word at index 2 is approximately 0.1857772922586975.

The TF-IDF matrix is presented in a compressed sparse row format, an efficient representation for sparse matrices. Each line in the output corresponds to a non-zero entry in the matrix, with the following components:

(i) (i, j) value: Represents a non-zero entry in the matrix

(ii) i: Row index

(iii) j: Column index

(iv) value: TF-IDF score for the term in the document.

6.1.2. Document similarity

Document similarity scores are computed using the TF-IDF matrix as a base. The matrix's non-zero entries show how crucial particular terms are for differentiating between papers. Comprehending the TF-IDF score distribution offers valuable perspectives on the distinct attributes of every document. Document similarities can be interpreted more nuancedly when terms with higher TF-IDF scores are identified as having substantially contributed to the document's content.

In the context of HMAC, this metric serves as a robust indicator of similarity between the uploaded assignment and other submissions. A high cosine similarity implies a closer resemblance between the two documents. The cosine similarity scores are stored in a list of tuples, where each tuple is structured as (currentfile_index, db_file_index, similarity_score).
where,

(i) currentfile_index: Refers to the index of the currently uploaded file

(ii) db_file_index: Refers to the index of a document stored in the database

(iii) similarity_score: Represents the calculated similarity between the current file and the database file.

Each tuple has three fields, with the 0^{th} index being the currently uploaded document, the first index being the document present in the database, and the second index being the similarity score. Below is an example of this list with two documents present in the database and uploading a third.

For better visualization, these scores can also be represented as a similarity matrix, where each cell at position (currentfile_index, db_file_index) contains the similarity score. The sample data is presented in Table 3. [(0, 1, 0.025970408434077174), (0, 2, 1.0), (1, 2, 0.025970408434077174)].

The diagonal entries are marked as "-" since a document's similarity with itself was not computed. This matrix representation complements the tuple format, making it easier to identify relationships between documents.

Table 3. Cosine similarity matrix representing the similarity scores between the current file and the database file

Current file/ Database file	File 1	File 2	File 3
File 1	-	0.025970	1.0
File 2	0.025970	-	0.025970
File 3	1.0	0.025970	-

6.1.3. Parallel comparison

ThreadPoolExecutor in Python facilitates concurrent task execution through threads, which is ideal for independent tasks like file similarity comparisons. It optimizes resource use by reusing threads and scales tasks efficiently by adjusting thread pool size. Asynchronous execution enables tasks to run independently, enhancing responsiveness. Compared to multiprocessing, which uses separate processes with higher memory overhead, and manual threading, which requires more complex thread management, ThreadPoolExecutor offers a simpler, more efficient solution. It is particularly advantageous for central processing unit-bound or I/O-bound tasks, balancing ease of implementation with performance gains in Python applications. Choosing ThreadPoolExecutor depends on task characteristics and integration needs within existing codebases and libraries. Given below is the algorithm for file comparison.

Here are the definitions for the variables and terms used in Algorithm 2:

- executor: The ThreadPoolExecutor instance used to manage concurrent task execution
- file_list: A list containing the file models to be compared against the uploaded file
- future_to_filename: A dictionary that maps each future task to its corresponding file model's filename
- comparisons: A list to store the comparison data, including the uploaded file, other files, and similarity results
- file: The current file being processed in the loop
- future: Represents a submitted task
- filename: Corresponds to a future task
- result: The result retrieved from a future task
- comparison_data: A dictionary that stores the comparison data for each file comparison
- Exception as e: The exception caught during error handling

Algorithm 2. File similarity comparison

```
function InitializeExecutor
    Create a ThreadPoolExecutor as executor
    return executor
end function

function SubmitTasks (executor, file_list,
future_to_filename)
    for each file in file list do
        future ← executor.submit (compare_file_
        similarity, file['content'], file['model'])
        future_to_filename[future] ← file['model']
    end for
end function

function ProcessCompletedTasks (futures, future_to_
filename, comparisons)
    for each future in futures that are completed do
    filename ← future_to_filename[future]
    result ← future.result()
    if result is valid then
        comparison_data ← {'uploaded file':uploaded_
        file_data, 'other_file': filname, 'similarity
        result':result}
        comparisons.append (comparison_data)
    else
        Error message for being unable to calculate
        similarity
    end if Exception as e
            Error message with exception details: e
  end for
end function

function Main (uploaded_file_data, file_list)
    executor ← InitializeExecutor
    future_to_filename ← {}
    SubmitTasks (executor, file_list,
    future_to_filename)
    comparisons ← []
    ProcessCompletedTasks (futures, future_to_
    filename, comparisons)
    Return comparisons
end function
```

7. Results

7.1. OCR

7.1.1. Document image generation

The primary purpose of this stage was to provide a standardized format for further analysis and to ensure uniformity in the subsequent stages of the project. The input at this stage is a single-page PDF file, as shown in Fig. 5.

7.1.2. Semantic segmentation with roboflow

This critical phase required the precise identification of individual words within the converted document images. This stage was critical in preparing the data for further processing, ensuring that the subsequent OCR phase focused on accurately segmented regions, allowing for more granular and precise extraction of text content from document images.

The properties of the segmented images, such as their x and y coordinates, width, and height, were systematically stored in the "box_dimensions" array. This array was critical in maintaining the spatial information of each word in the document.

For Fig. 5, the following shows its "box_dimension" array.

[{'index': 1, 'x': 602, 'y': 360, 'width': 102, 'height': 39},

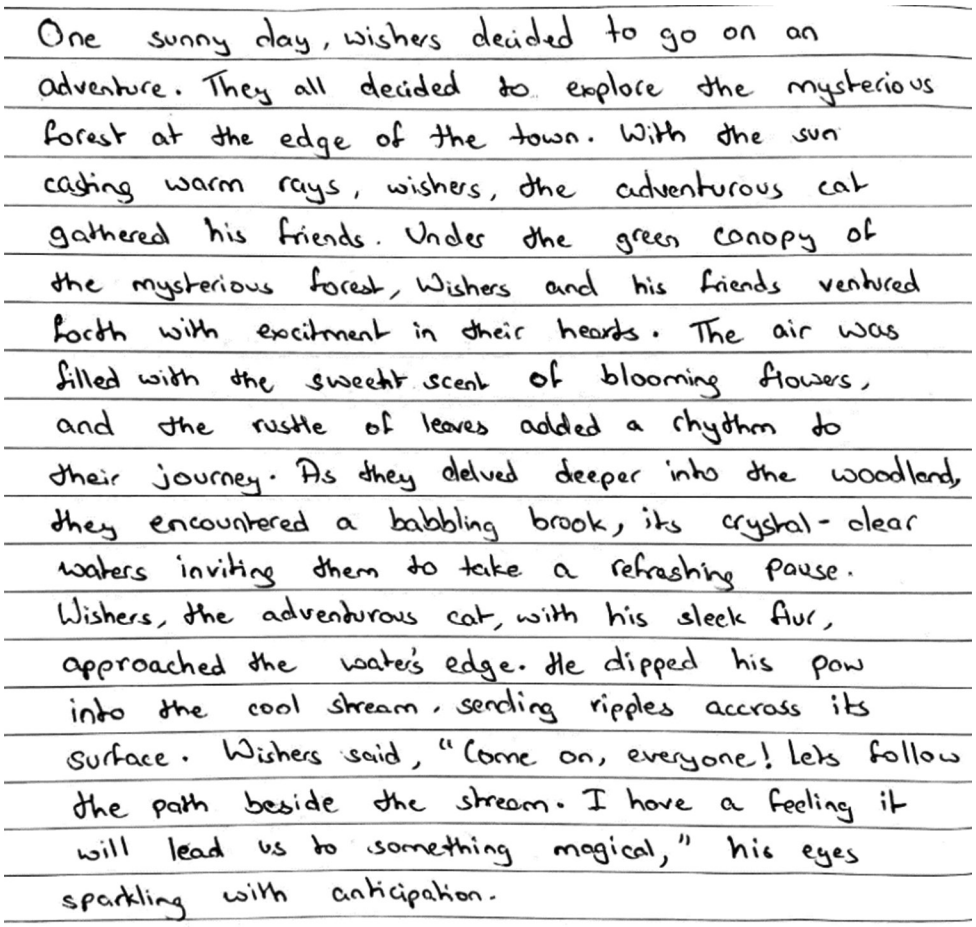

One sunny day, wishers decided to go on an adventure. They all decided to explore the mysterious forest at the edge of the town. With the sun casting warm rays, wishers, the adventurous cat gathered his friends. Under the green canopy of the mysterious forest, Wishers and his friends ventured forth with excitment in their hearts. The air was filled with the sweetht scent of blooming flowers, and the rustle of leaves added a rhythm to their journey. As they delved deeper into the woodland, they encountered a babbling brook, its crystal-clear waters inviting them to take a refreshing pause. Wishers, the adventurous cat, with his sleek fur, approached the water's edge. He dipped his paw into the cool stream, sending ripples accross its surface. Wishers said, "Come on, everyone! Lets follow the path beside the stream. I have a feeling it will lead us to something magical," his eyes sparkling with anticipation.

Fig. 5. Document image generation: standardizing input for subsequent stages. This image represents the outcome of the first stage in the optical character recognition workflow, which focuses on document image generation

{'index': 2, 'x': 33, 'y': 144, 'width': 86, 'height': 35},
{'index': 3, 'x': 691, 'y': 706, 'width': 79, 'height': 34},
….
{'index': 154, 'x': 559, 'y': 372, 'width': 23, 'height': 14},
{'index': 155, 'x': 390, 'y': 660, 'width': 17, 'height': 14},
{'index': 156, 'x': 608, 'y': 747, 'width': 15, 'height': 13}]

The processed words were systematically organized and saved in a specific folder. Each word was given a unique title, denoted as "words_Index," with the index corresponding to the word's position in the "box_dimensions" array. The table 4 depicts one such example Roboflow model's output. This indexing system provided a seamless way to access the associated images by directly referencing the "box_dimensions" array.

7.1.3. Lexical ordering algorithm

This stage addressed the challenge of ensuring the correct sequence of words within the document, particularly when dealing with unordered or randomly detected words. The algorithm aimed to arrange the words in a sequence that accurately reflected the original order in the document.

After applying this algorithm, the "box_dimension" array for Fig. 6 is updated as follows:

[{'index': 18, 'x': 25, 'y': 9, 'width': 65, 'height': 32},
{'index': 117, 'x': 126, 'y': 19, 'width': 80, 'height': 27},
{'index': 74, 'x': 223, 'y': 7, 'width': 74, 'height': 37},
...
{'index': 78, 'x': 52, 'y': 793, 'width': 113, 'height': 35},
{'index': 82, 'x': 195, 'y': 790, 'width': 64, 'height': 23},
{'index': 126, 'x': 293, 'y': 787, 'width': 155, 'height': 34}]

The system's systematic word indexing allows for quick access to individual words by index number. For example, if a word is given the index number 10, it indicates that it is the first word in the corresponding PDF section. Based on the array after applying the algorithm to Fig. 5, the ordered words are produced as depicted in Table 5.

Table 4. Word detection and indexing results. This table displays the Roboflow model's output, demonstrating the detection of words in random order across various images

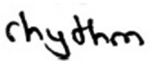
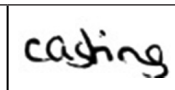
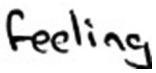
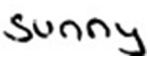
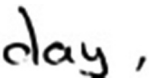
Image			
Title (words_Index)	words_1.jpg	words_2.jpg	words_3.jpg

Table 5. Ordered words extracted after applying the lexical ordering algorithm. This table presents the first three words extracted in order after applying the lexical ordering algorithm (Section 4.3)

Image			
Title (words_Index)	words_18.jpg	words_117.jpg	words_74.jpg

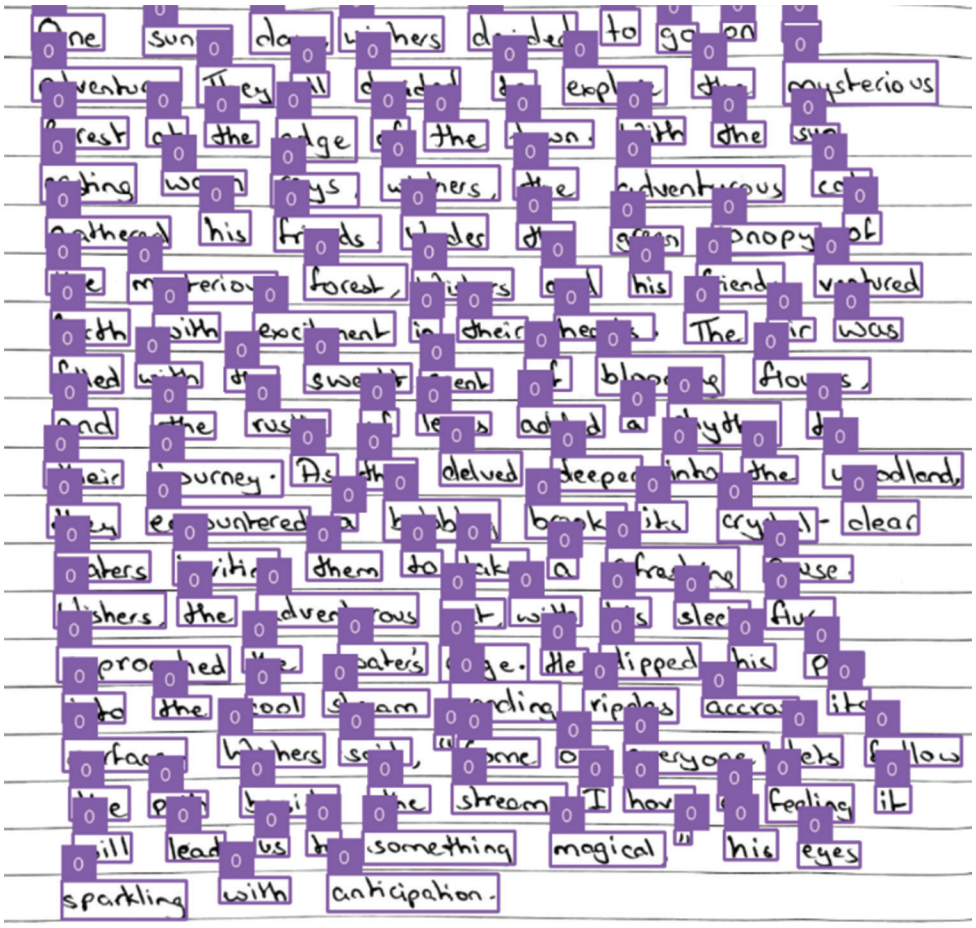


Fig. 6. Semantic segmentation with Roboflow—refined output. This image shows the results of the semantic segmentation phase using Roboflow

7.1.4. Image composition module

Rather than processing each word image individually, this method involved concatenating groups of 10-word images into a single, continuous line.

For Fig. 5, this stage produced the following output (Fig. 7), based on a batch size of 11.

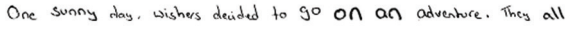


Fig. 7. Image composition module output: optimized concatenation. This image depicts the output of the image composition module (Section 7.1.4). It displays the first 11 words, as the specified batch_size was 11, demonstrating the effectiveness of the optimized concatenation approach

7.1.5. Deep learning model inference

The image composition module generated concatenated lines, which were then fed into the OCR model for recognition and transcription. Using the transformer's encoder-decoder structure, the model successfully converted visual features extracted from images into a coherent sequence of word pieces, effectively reconstructing the original handwritten content.

For Fig. 6, the model generates the following output:

One sunny day, wishes decided to go on ACT Adventure. They all decided to explore the mysterious forest at the edge of the town with the SWR. Casting warm rays. Wishers, the adventurous cat gathered his friends. Under the Green Canopy of the mysterious forest, Wishers and his friends ventured forth with excitement in their hearts. The air was filled with the sweet scent of blooming flowers, and the rustle of leaves added a rhythm to their journey. As they delved deeper into another woodland, they encountered a balding brook, its crystal clear waters inviting them to take A refastening pause. wishers, the adventurous cat, with his sleek fur, approached the water's edge. He dipped his POW into the cool stream sending rifles accrabs its surface Wishers said "Come On, everyone! Let us follow the path beside the stream. I have A feeling it will lead the US to something magical, It his eyes sparkling with anticipation.

7.1.6. Evaluation

To evaluate the results, we used three error measurements: character error rate (CER), word error rate (WER), and word recognition accuracy (WRA). These evaluation metrics is replicating the performance indices as mentioned in Appendix 1.

The CER, or percentage of erroneous characters in the system output, is a common metric in OCR tasks. It can be computed by dividing the number of incorrect characters by the sum of correct characters and errors in the system output. Similarly, the WER represents the percentage of incorrect words in the system output. It is computed by dividing the number of incorrect words by the sum of correct words and system errors.

$$\text{CER, WER} = \frac{\text{Errors}}{\text{Correct} + \text{Errors}} \tag{5}$$

Similar to WER, WRA measures the accuracy of whole-word recognition. It is calculated as the ratio of correctly recognized words to the total number of words.

$$\text{WRA} = \frac{\text{Correctly recognized words}}{\text{Total words}} \tag{6}$$

To determine the number of errors, we first aligned the ground truth sentence and OCR prediction lines at the character level (both CER and WER). We then calculated the overall Levenshtein distance between the system output and the ground truth, considering deletions and insertions (Li et al., 2021).

While calculating the CER is relatively simple, different evaluation systems employ different alignment approaches when calculating WER. Fig. 8 illustrates one alignment of a misspelled word example. The alignment is character-level, so the missing letters "e" and "x" will be paired with the empty string. If this alignment is used to calculate WER, the word example will be paired with the entire word exam, resulting in one error.

7.1.6.1. Handwriting quality assessment

It is important to note that the experiment used the same type of handwriting for all qualities. For instance, to maintain consistency in the evaluation, "Good" handwriting only includes text from one individual.

7.1.6.2. Factors affecting handwriting quality

To assess any handwriting, it is essential to know the elements that affect handwriting quality. Every element, from word spacing to writing style, influences our system's overall efficacy in a variety of handwriting attributes. Table 6 presents the factors that are taken into consideration for handwriting quality.

7.1.6.3. Word-level error analysis

Word-level error analysis is a thorough examination of our handwriting recognition system's

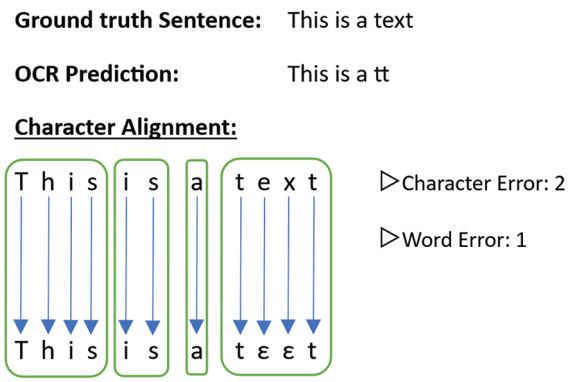


Fig. 8. An example of how the number of word errors varies according to alignment. After aligning the lines at the character level, aligning them at the word level results in a word error count of 1 (e.g., the word example is aligned with t ε ε t, the empty string)
Abbreviation: OCR: Optical character recognition

performance at the level of individual words. We evaluated the recognition accuracy of each sample by comparing the actual text to the system's output. The CER and WER were calculated to assess the system's ability to correctly identify individual characters and entire words. In Table 7, the CER and WER percentages are shown.

7.1.6.4. Line-level error analysis

By evaluating the recognition accuracy of complete lines of text, line-level error analysis provides a more comprehensive view than just analyzing individual words. Each sample's real lines and those identified by the system were compared. We evaluated the whole line recognition accuracy using line CER and line WER metrics.

An error analysis was performed at the line level. We examined how many lines were recognized completely correctly and found that most of the lines did not have any errors. With an average of over 70%, every 12 lines were recognized correctly. In the set of incorrect lines, most contained only minor errors, typically due to common OCR confusion as described by Kissos and Dershowitz (2016), Levenshtein, (1966).

7.1.6.5. Evaluation table

In this comprehensive performance evaluation, we present the results of our HMAC system across a range of handwriting qualities. We designed a structured table with important indicators to provide a more detailed picture of our system's functioning. The table includes an analysis of the number of sentences, handwriting quality (best, good, and worst), total lines processed, processing time, average speed per sentence, and accuracy score.

Table 7 shows a notable relationship between handwriting quality and OCR performance metrics. As handwriting declines from "Best" to "Worst," processing time, average speed per sentence, and error rates (both CER and WER) increase significantly, while WRA decreases dramatically. Specifically, "Best" handwriting quality has the lowest CER and WER rates, 2.35% and 8.11%, respectively, and the highest WRA at 95.30%. In contrast, "worst" handwriting quality results in a significantly higher CER (33.22%) and WER (66.44%), with a drastic reduction in WRA to as low as 4.03%. This degradation is reflected in the average speed per sentence, which rises from around 22 seconds for "Best" handwriting to more than 39 seconds for "Worst." These patterns indicate that as

Table 6. Summary of key criteria for evaluating handwriting quality

Handwriting	Consistency in Y-coordinate	Spacing between words	Character legibility	Pen pressure and stroke consistency	Aesthetic appeal
Best	Same line	Consistent	Clearly legible characters	Uniform throughout the writing	Pleasing
Average	Some variation, within a defined threshold	Minor variations, within an acceptable range	Some variations not hinder overall legibility	Minor variations	Adequate with room for improvement
Worst	Significant variation	Irregular or excessive	Inconsistencies	Significant irregularities	Unattractive or messy

Table 7. Handwriting Match and Artificial Intelligence Content System Performance Evaluation

Number of sentences	3 (43 words, including punctuation)			6 (115 words, including punctuation)			11 (181 words, including punctuation)			Average (6.6 words)
Handwriting qualities	Best	Good	Worst	Best	Good	Worst	Best	Good	Worst	Average
Total lines	3	5	7	9	12	15	14	19	22	11.7
Processing time (seconds)	65.07	71.8	80.6	134.76	286.85	216.90	414.4	414.9	430.7	244.97
Average speed per sentence (seconds)	21.69	23.93	26.8	22.46	47.8	36.15	37.67	37.71	39.15	37.15
CER (%)	2.35	4.69	17.92	1.18	4.04	28.74	2.03	7.14	33.22	8.56
WER (%)	8.11	21.62	40.54	5	12	68.00	8.97	19.87	66.44	24.22
WRA (%)	95.30	83.70	10.81	95.60	83	2.00	78.21	53.85	4.03	57.76

Abbreviations: CER: Character error rate; WER: Word error rate; WRA: Word recognition accuracy.

handwriting clarity deteriorates, OCR systems struggle more, taking longer to process text and producing lower accuracy. This emphasizes the importance of handwriting quality in achieving efficient and accurate OCR, demonstrating that even minor deviations from optimal handwriting can have a significant impact on recognition performance.

The graphical representation of Table 7 offers a clear visual insight into our OCR system's performance across various handwriting qualities. This graphical representation facilitates an intuitive understanding of the system's capabilities and limitations by highlighting trends and disparities across various categories of handwriting quality. Fig. 9 shows distinct trends by plotting a weighted sum of key metrics: total lines, processing time (seconds), average speed per sentence (seconds), CER, WER, and WRA. The bar colors represent different sentence counts—3 sentences (blue), 6 sentences (green), and 11 sentences (red), while the black line represents the average sentence count for each metric.

7.1.7. Limitations

Handwriting recognition through OCR encounters several challenges, impacting its effectiveness in capturing and interpreting diverse styles of handwritten text. It struggles with text written directly on lines, as this deviates from the standard way of writing. In contrast, it performs very well when text is placed between lines. Furthermore, OCR's reliance on perfectly aligned images makes it vulnerable to inaccuracies with tilted or rotated inputs. Page curvature complicates recognition because OCR is designed for flat pages and can struggle with curved surfaces. Inconsistencies in lined pages, as well as deviations from expected straight-line formation, have an impact on accuracy. A fixed threshold for recognizing handwriting does not account for the variability in individual styles, and poor handwriting can result in time-consuming processing, reducing real-time efficiency. It is obvious that additional developments in OCR technology are required considering these constraints. It is imperative to tackle these challenges to maximize the practical uses of OCR and enhance its flexibility to the ever-changing scenarios posed by handwritten documents in the real world.

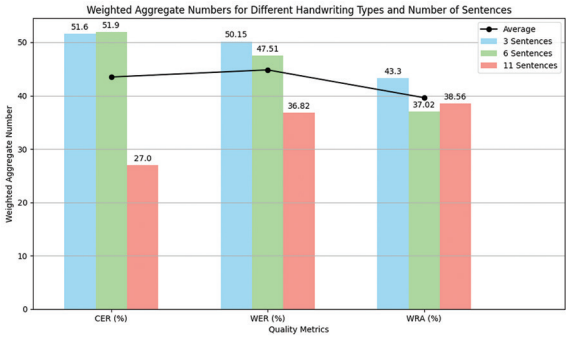


Fig. 9. Graphical representation of Table 7, offering clear visual insight for the untrained eye

Table 8. Performance measures before and after fine-tuning. This table compares key performance measures, such as accuracy and F1 score, before and after the fine-tuning process

Performance measures	Before fine-tuning	After fine-tuning
Accuracy	93.07	98.3
F1	0.68	0.84

7.2. AI Content Detection

The trained AI model achieved commendable accuracy in distinguishing between human-generated and AI-generated text, as shown in Table 7. After a meticulous training process on a dataset tailored to the study's objectives, the model demonstrated a strong ability to flag content with the signature characteristics of AI-generated language.

The OCR stage extracts textual content from images, which is then used as input for the AI content detection model within the HMAC system. The AI content detection model, which is primarily based on advanced transformer-based architecture, examines the provided text to determine whether it was generated by a human or by AI. The findings of this detection process are complex and offer insightful information about the text in question. The model calculates the percentage of content that is attributed to human authorship and the percentage that has the characteristics of AI-generated language. The output's dual nature serves as a quantifiable breakdown, clearly showing the proportion of the submitted assignments generated by AI versus those written by humans. The text shown in Fig. 7 was generated using ChatGPT, and when processed by the model, it produced the following result.

- Class 0: 01.20%
- Class 1: 98.80%.

7.3. Performance Metrics

The performance of the AI content detection model was evaluated at two stages: before and after fine-tuning. The model was refined, allowing it to distinguish between human-generated and AI-generated content more accurately. The model's remarkable accuracy of 93.07% before fine-tuning demonstrates its capacity to accurately classify the source of content in each

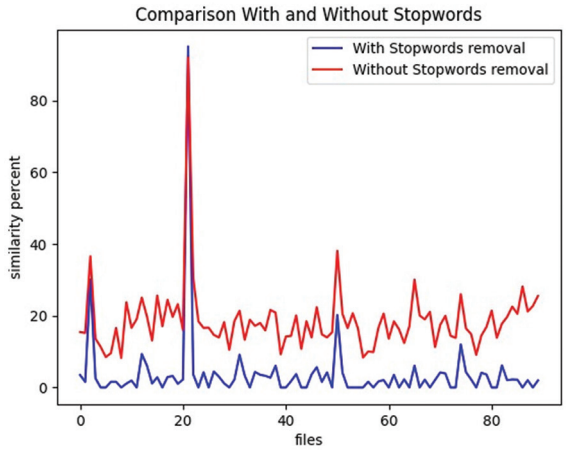


Fig. 10. Comparison of similarity percentage with and without stopwords

assignment. After applying fine-tuning techniques, its accuracy dramatically increased to 98.3%. Moreover, significant improvements were observed in the F1 score, a measure that strikes a balance between recall and precision. Before fine-tuning, the F1 score was 0.68, indicating a satisfactory performance. After fine-tuning, it rose dramatically to a remarkable 0.84. The result are well presented in Table 8.

7.4. Similarity Check

For similarity detection, the text generated from the handwriting detection step was compared to the previously submitted assignments stored in the database.

Fig. 10 depicts the change in similarity detection results before and after removing stopwords.

For this graph, a total of 21 files were used for comparison. The contents of 10 files were completely unique, while the remaining files contained similar to the first set of 10 files. The graph clearly shows that removing stopwords significantly decreased the similarity percentage. The mean difference in similarity was calculated to be around 14.905 using the specified formula.

8. Conclusion and Recommendations

This multiple-layered strategy demonstrates that HMAC is a useful platform that not only detects the use of AI-generated content but also actively addresses issues on academic integrity. The technology provides students with an easy platform to submit assignments, streamlining the assessment process for professors. By doing so, HMAC provides teachers with the resources they need to properly detect duplicate and AI-generated content, ensuring an impartial and rigorous assessment environment. HMAC serves as a preventive measure against the inappropriate use of generative models in educational contexts and is crucial in preserving the learning objectives of assignments by providing teachers with a useful tool for assessment. It helps to maintain the educational value of assignments while actively discouraging the misuse of generative models. HMAC acts as a vital safety net as education evolves in the digital age, ensuring that assignments fulfill their purpose of fostering authentic learning experiences.

The contributions of this paper include the following:

(i) Improved handwritten content extraction with OCR: This paper describes how our OCR component improved the extraction and interpretation of handwritten content from images, particularly in the context of academic documents and assignments. This includes improvements in accuracy and efficiency, which add to the overall landscape of document digitization.

(ii) Enhanced identification of AI-generated content: Compared to other approaches, this research demonstrates how HMAC considerably enhances the recognition of AI-generated content while upholding academic integrity.

Expanding the system's support for multi-page PDFs is a critical priority, necessitating optimizations in the OCR, content detection, and plagiarism-checking modules to enable seamless navigation of assignments spanning multiple pages. The applicability of HMAC to a wider range of academic resources would be significantly increased through this modification. The future holds promise for improving HMAC's understanding of semantic context and enabling more nuanced contextual analysis. Incorporating sophisticated natural language processing techniques would allow the system to not only identify content but also comprehend its meaning, fostering a better understanding of assignments and their legitimacy. Implementing a strong user feedback mechanism is an important part of future development. Allowing users to provide feedback on AI content detection accuracy and plagiarism checks promotes iterative refinement. A more user-centric and efficient system can be created with the assistance of user-generated suggestions for handling particular assignment types or enhancing the user interface. Furthermore, the integration of HMAC with Learning Management Systems shows promise for streamlining assignment submission and analysis processes in educational institutions. Developing plugins or application programming interfaces for seamless integration with Moodle, Canvas, and Blackboard, among other educational platforms, improves HMAC's accessibility and usability in educational ecosystems.

In addition, incorporating features that provide users with insights into the decision-making processes of AI content detection models, thereby increasing transparency and trust, is an important consideration for future iterations of HMAC. This comprehensive vision for future development establishes HMAC as an evolving, adaptable, and user-friendly system at the forefront of content analysis and plagiarism detection in educational settings.

References

Bhat, A. (2023). *GPT-Wiki-Intro (Revision 0e458f5). Hugging Face*. Available from: https://huggingface.co/datasets/aadityaubhat/gpt-wiki-intro [Last accessed on 2024 May 24].

Daniel, F., Cappiello, C., & Benatallah, B. (2019). *Bots Acting Like Humans: Understanding and Preventing Harm*. Available from: https://www.floriandaniel.it/papers/danielic2019.pdf [Last accessed on 2024 May 24].

Dong, R., & Smith, D.A. (2018). Multi-input attention for unsupervised OCR correction. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*. Vol. 1. p2363–2372.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of deep bidirectional transformers for language understanding. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*: Human Language Technologies, Volume 1 (Long and Short Papers), 4171–4186.

Englmeier, T., Fink, F., & Schulz, K.U. (2019). AI-PoCoTo-combining automated and interactive OCR postcorrection. In: *Proceedings of the Third International Conference on Digital Access to Textual Cultural Heritage*. ACM.

Evershed, J., & Fitch, K. (2014). Correcting noisy OCR: Context beats confusion. In: *Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage*. ACM, p45–51.

Graves, A., Fernández, S., Gomez, F., & Schmidhuber, J. (2006). Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In: *Proceedings of the 23rd International Conference on Machine Learning*. ACM, p369–376.

Guha, R., Das, N., Kundu, M., Nasipuri, M., & Santosh, K. (2019). Devnet: An efficient cnn architecture for handwritten Devanagari character recognition. In: *International Journal of Pattern Recognition and Artificial Intelliegence*. World Scientific, Singapore.

Hämäläinen, M., & Hengchen, S. (2019). From the paft to the fiiture: A fully automatic NMT and word embeddings method for OCR post-correction. In: *Proceeding of International Conference on Recent Advances in Natural Language Processing*. INCOMA, p432–437.

Jain, A.K., & Yu, B. (1998). Automatic text location in images and video frames. In: *Proceeding of International Conference of Pattern Recognition.* ICPR, Brisbane, p1497–1499.

Jauhiainen, T.S., Linden, B.K.J., & Jauhiainen, H.A. (2016). Heli, a word-based backoff method for language identification. In: *Proceedings of the Third Workshop on NLP for Similar Languages Varieties and Dialects VarDial3*. Osaka, Japan, p12.

Kauppinen, P. (2016). *OCR Post-Processing by Parallel Replace Rules Implemented as Weighted Finite-State Transducers*. University of Helsinki, Finland.

Kettunen, K., & Koistinen, M. (2019). *Open Source Tesseract in re-OCR of Finnish Fraktur from 19th and Early 20th Century Newspapers and Journals-Collected Notes on Quality Improvement*. Digital Humanitarian Network, Virtual, p270–282.

Kettunen, K., Kervinen, J., & Koistinen, M. (2018). Creating and using ground truth OCR sample data for Finnish historical newspapers and journals. In: *Proceeding of DHN 2018 Digital Humanities in the Nordic Countries 3rd Conference*. Helsinki.

Kim, P.K. (1999). *Automatic Text Location in Complex Color Images Using Local Color Quantization*. Vol. 1. IEEE TENCON, p629-632.

Kissos, I., & Dershowitz, N. (2016). OCR error correction using character correction and feature-based word classification. In: *2016 12th IAPR Workshop on Document Analysis Systems (DAS)*. IEEE, p198–203.

Koistinen, M., Kettunen, K., & Kervinen, J. (2017). How to improve optical character recognition of historical finnish newspapers using open source tesseract OCR engine? In: *Proceedings of the LTC*. p279–283.

Koistinen, M., Kettunen, K., & Pääkkönen, T. (2017). Improving optical character recognition of finnish historical newspapers with a combination of fraktur and antiqua models and image preprocessing. In: *Proceedings of the 21st Nordic Conference on Computational Linguistics*. p277–283.

Levenshtein, V.I. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10, 707–710.

Li, H., & Doermann, D. (1998). Automatic text tracking in digital videos. In: *Proceeding of IEEE 1998*

Workshop on Multimedia Signal Processing, Redondo Beach, California, USA, p21–26.

Li, M., Lv, T., Cui, L., Lu, Y., Florencio, D., Zhang, C., Li, Z., & Wei, F. (2021). *TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models*. arXiv, 2109, 10282.

Lindén, K., Silfverberg, M., Pirinen, T., Hardwick, S., Drobac, S., & Axelson, E. (2012). *HFST-An Environment for Creating Language Technology Applications*. *Studies in Computational Intelligence*. Springer, Berlin.

Llobet, R., Cerdan-Navarro, J.R., Perez-Cortes, J.C., & Arlandis, J. (2010). OCR post-processing using weighted finite-state transducers. In: *2010 20th International Conference on Pattern Recognition*. p2021–2024.

Lu, N., Liu, S., He, R., Wang, Q., Ong, Y.S., & Tang, K. (2024). *Large Language Models can be Guided to Evade AI-Generated Text Detection*. https://doi.org/10.48550/arXiv.2305.10847

Lund, W.B., Kennard, D.J., & Ringger, E.K. (2013). Combining multiple thresholding binarization values to improve OCR output. In: *Document Recognition and Retrieval XX*, Vol. 8658. International Society for Optics and Photonics, p86580R.

Lund, W.B., Walker, D.D., & Ringger, E.K. (2011). Progressive alignment and discriminative error correction for multiple OCR engines. In: *2011 International Conference on Document Analysis and Recognition*. IEEE, p764–768.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient Estimation of Word Representations in Vector Space*. https://doi.org/10.48550/arXiv.1301.3781

Mindner, L., Schlippe, T., & Schaaff, K. (2023). *Classification of Human- and AI-Generated Texts: Investigating Features for ChatGPT*. arXiv, 2308, 05341.

Mitrović, S., Andreoletti, D., & Ayoub, O. (2023). *ChatGPT or Human? Detect and Explain. Explaining Decisions of Machine Learning Model for Detecting Short ChatGPT-generated Text*. https://doi.org/10.48550/arXiv.2301.13852

Pelau, C., Dabija, D.-C., & Ene, I. (2021). *What makes an AI device human-like? The role of interaction quality, empathy and perceived psychological anthropomorphic characteristics in the acceptance of artificial intelligence in the service industry. Computers in Human Behavior*, 122, 106855.

Reul, C., Christ, D., Hartelt, A., Balbach, N., Wehner, M., Springmann, U., Wick, C., Grundig, C., Büttner, A., & Puppe, F. (2019). *Ocr4all-an Open-Source Tool Providing a (Semi-) Automatic OCR Workflow for Historical Printings*. https://doi.org/10.48550/arXiv.1909.04032

Reul, C., Springmann, U., Wick, C., & Puppe, F. (2018). *State of the art optical character recognition of 19th century Fraktur scripts using open source engines*. https://doi.org/10.48550/arXiv.1810.03436

Reynaert, M.W. (2010). Character confusion versus focus word-based correction of spelling and OCR variants in corpora. *International Journal of Documents Analysis and Recognition (IJDAR),* 14(2), 173–187.

Rodriguez, J.D., Hay, T., Gros, D., Shamsi, Z., & Srinivasan, R. (2022). *Cross-Domain Detection of GPT-2-Generated Technical Text*. Available from: https://aclanthology.org/2022.naacl-main.88 [Last accessed on 2024 May 24].

Romero, V., Toselli, A.H., & Vidal, E. (2012). *Multimodal Interactive Handwritten Text Transcription*. Vol. 80. World Scientific, Singapore.

Sabu, A. M., & Das, A. S. (2018). A survey on various optical character recognition techniques. *In Proceedings of the 2018 International Conference on Emerging Devices and Smart Systems* (ICEDSS) (pp. 1–5). IEEE. https://doi.org/10.1109/ICEDSS.2018.8544323

Sadasivan V.S., Kumar, A., Balasubramanian, S., Wang, W., & Feizi, S. (2023). *Can AI-Generated Text be Reliably Detected?* https://doi.org/10.48550/arXiv.2303.11156

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). *DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter*. https://doi.org/10.48550/arXiv.1910.01108

Silfverberg, M., Kauppinen, P., & Lindén, K. (2016). Data-driven spelling correction using weighted finite-state methods. In: *Proceedings of the SIGFSM Workshop on Statistical NLP and Weighted Automata*. Association for Computational Linguistics, Berlin, p51–59.

Springmann, U., & Lüdeling, A. (2016). *OCR of Historical Printings with an Application to Building Diachronic Corpora: A Case Study Using the RIDGES Herbal Corpus*. https://doi.org/10.48550/arXiv.1608.02153

Springmann, U., Najock, D., Morgenroth, H., Schmid, H., Gotscharek, A., & Fink, F.(2014). OCR of historical printings of latin texts: Problems, prospects, progress. In: *Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage*. ACM, p71–75.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1),

1929–1958.

Uzun, L. (2023). *ChatGPT and Academic Integrity Concerns: Detecting Artificial Intelligence Generated Content*. Available from: https://www.researchgate.net/publication/370299956-chatgpt-and-academic-integrity-concerns-detecting-artificial-intelligence-generated-content [Last accessed on 2024 May 24].

Vobl, T., Gotscharek, A., Reffle, U., Ringlstetter, C., & Schulz, K.U. (2014). Pocoto-an open source system for efficient interactive postcorrection of OCRed historical texts. In: *Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage*. ACM, p57–61.

Wahle, J. P., Ruas, T., Mohammad, S. M., Meuschke, N., & Gipp, B. (2023). *AI Usage Cards: Responsibly reporting AI-generated content* [Conference poster]. 2023 ACM/IEEE Joint Conference on Digital Libraries (JCDL), Santa Fe, NM, USA. https://doi.org/10.1109/JCDL57899.2023.00060

Wick, C., Reul, C., & Puppe, F. (2018). *Calamari-a High-Performance Tensorflow-Based Deep Learning Package for Optical Character Recognition*. https://doi.org/10.48550/arXiv.1807.02004

Wick, C., Reul, C., & Puppe, F. (2018). Comparison of OCR accuracy on early printed books using the open source engines Calamari and OCRopus. *Journal for Language and Conputational Linguistics*, 33, 79–96.

Wu, V., Manmatha, R., & Riseman, E.M. (1997). Finding text in images. In: *Proceedings of Second ACM International Conference on Digital Libraries*. Philadelphia, PA,p23–26.

AUTHOR BIOGRAPHY

Hrishikesh Pramod Panigrahi is currently pursuing a Bachelor of Engineering in Computer Engineering at The Bombay Salesian Society's Don Bosco Institute of Technology, Mumbai University. He has demonstrated strong technical expertise through hands-on experience in backend development, machine learning, and web applications. Hrishikesh has led backend teams, contributed to AI-driven academic integrity tools, and developed innovative mobile and web solutions. His technical proficiency includes programming languages such as Java, GoLang, Python, and Kotlin, and frameworks such as Django, ReactJS, and TensorFlow. He has published a research paper titled "Smart Posture Analyzer For Exercise" at the Institute of Electrical and Electronics Engineering (IEEE). His areas of interest include artificial intelligence, system design, and full-stack development.

Siddhanth Naidu is a Computer Engineering student at The Bombay Salesian Society's Don Bosco Institute of Technology, Kurla (Mumbai University). He has worked on impactful projects such as FLMS (Flood Location, Management, and Solution), an IEEE-published flood management system, and Dawn of Survival, an Android game developed in Unreal Engine. His technical skills include Java, Spring Boot, Python, ReactJS, HTML/CSS, Unreal Engine, and Unity. He has presented research at the International Conference on IoT, Communication, and Automation Technology (ICACAT) 2023 and was part of the ACM (Aluminum Composite Material) Design Team at Don Bosco Institute of Technology, Kurla. Siddhanth's interests lie in AI, web development, system design, and game design.

Ambuj Pandey is a Computer Engineering graduate from The Bombay Salesian Society's Don Bosco Institute of Technology, Kurla (Mumbai University). He has worked on impactful projects such as EDT, Expiry (Expiry Date Tracker, an Android app to track the expiry of purchased groceries), FLMS (an IEEE-published flood management system), and Dawn of Survival (an Android game developed in Unreal Engine) during his college years. His technical skills include C++, Python, Docker, Django, ReactJS, HTML/CSS, and Unreal Engine. He has published a research paper at ICACAT 2023. Ambuj's interests lie in AI, web development, and game design.

Phiroj Shaikh is currently an Associate Professor at the Department of Computer Engineering, The Bombay Salesian Society's Don Bosco Institute of Technology, Mumbai, India, affiliated with the University of Mumbai. He earned a PhD degree in Computer Science & Technology (in the domain of Web Data Mining) from Nagpur University, Nagpur, India. He has academic experience of over two decades with more than 60 research publications in journals/conferences. His areas of interest include natural language processing (with a special focus on regional language development), computational linguistics, and algorithm analysis. He is currently focusing on Educational Technology research area. He has been invited as an expert for faculty development programs, as a session expert, and as a reviewer at various conferences.

Amiya Kumar Tripathy is currently a Professor at the Department of Computer Engineering, The Bombay Salesian Society's Don Bosco Institute of Technology, Mumbai, India, affiliated with the University of Mumbai. He earned a PhD degree in Computer Science & Engineering (in the domain of Data Mining & Wireless Sensor Networks) from the Indian Institute of Technology Bombay, Mumbai, India. He was an adjunct professor at the School of Science, Edith Cowan University (ECU), Australia. He was a visiting researcher at the Rajamangala University of Technology, Bangkok, Thailand, working on Internet of Things (IoT)-enabled remote monitoring for the Precision Agriculture Farming project. He has been in the software industry, research, and academia for more than two decades, having around 150 publications in journals/conference papers. His research focuses on data science, computer vision, remote sensing, and IoT for Precision Agriculture. He has contributed to numerous collaborative research and consultancy projects in the domain of data analytics in India and abroad. He has served on the technical program committees of several international conferences, been invited as a plenary speaker, and co-chaired sessions at various conferences.

Appendix

Appendix 1. Explanation of Precision, Recall, and F1-score

In this appendix, we provide an explanation of precision, recall, and F1-score, which are used as evaluation metrics for HMAC.

(i) *Precision:* A measure of the accuracy of positive predictions. It is defined as the ratio of true positives to the sum of true positives and false positives.

$$\text{Precision} = \frac{\text{No. of true positives}}{\text{No. of true positives} + \text{No. of false positives}}$$

A high precision indicates a small number of false positives, meaning that the model has a low tendency to classify negative instances as positive.

(ii) *Recall:* Also known as sensitivity or the true positive rate, it measures the proportion of actual positive instances that are correctly identified by the model. It is defined as the ratio of true positives to the sum of true positives and false negatives.

$$\text{Recall} = \frac{\text{No. of true positives}}{\text{No. of true positives} + \text{No. of false negatives}}$$

A high recall indicates a small number of false negatives, meaning that the model effectively captures most of the positive instances.

(iii) *F1-score:* The F1-score is the harmonic mean of precision and recall, providing a balanced measure of both metrics. It can be computed using the following formula:

$$F1 - score = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

The F1-score ranges between 0 and 1, where a value of 1 represents a perfect balance between precision and recall.