Detecting AI-Generated Text: A Comparative Study of Machine Learning and Deep Learning Approaches

Diaa Salama AbdElminaam[1], Adham Mohamed[2], Ammar Alaa[3],
Youssef Mohamed Abdelshahid[4], Youssef Mohamed Amer [5]
Faculty of Computer Science
Misr International University, Cairo, Egypt
diaa.salama[1], adham2204058[2],
ammar2201210[3], youssef2205890[4], youssef2207525[5]{@miuegypt.edu.eg}

***Abstract*—The rapid development of AI text generation tools such as Chat-GPT, Gemini, and other LLMs have had a large impact on the originality and inspiration of today's work, it is now very common for anyone to use AI to generate his work whether its text or other endeavors. Our aim in this paper is to provide a detailed and wide variety of ways AI generated text could be detected. Moreover, in analyzing AI generated text, we use two different languages: Arabic and English. We introduce two major components: Firstly, using machine learning algorithms and models like Random Forests, Logistic Regression, Naive Bayes, XGBoost, and Passive Aggressive classifier, with the TF-IDF Tokenizer acting as feature extraction. Secondly, using deep learning algorithms, these include: Convolutional Neural Network (CNN), Deep Neural Network (DNN), Gated Recurrent Unit (GRU), and Bidirectional Long Short-Term Memory Neural Network (Bi-LSTM). In addition, we tested two pre-trained transformer models from google, these being, ELECTRA and RoBERTa. The models were equipped with evaluation metrics like accuracy, f1-score, recall, support, etc. This was to show, measure and compare the performance of these algorithms and models. These experiments yielded impressive results and proposed that Deep Learning models accomplished the finest results and accuracy.**

Keywords: Convolutional Neural Network (CNN), Deep Neural Network (DNN), Gated Recurrent Unit (GRU), Bidirectional Long Short-Term Memory Neural Network (Bi-LSTM), Random Forest, Logistic Regression, Naive Bayes, XGBoost, and Passive Aggressive classifier, TF-IDF

I. Introduction

As Language models and artificial intelligence continue to evolve, they are now able to generate text that closely reflects human writing in language, and structure. In areas such as content creation, automation, and personalized communication, this has led to amazing opportunities. It has, however, brought up challenging problems—especially with regards to originality, and authorship. Especially in fields such as education, journalism, and scientific publishing, where the authenticity of written material serves as a measure of judgment, the need to separate human-written from machine-authored text has never been more relevant. Thus, such reliable detection tools contribute not only to maintaining ethical principles but also to saving people's trust in communication in a world where AI-generated material has become a part of daily life and is now accessible to everyone.

Despite the growing availability of AI-detection tools, many struggle with consistency and reliability, especially when faced with nuanced or well-structured machine-generated text. A recent study [1] demonstrated how current tools can mislabel authentic academic writing as AI-generated, while also failing to confidently detect text created entirely by large language models. These inconsistencies pose a real risk to academic fairness and trust in automated detection systems. Beyond this, a broader challenge is emerging: as generative AI becomes more embedded in everyday workflows, the lines between human and machine collaboration are blurring. This makes it increasingly difficult to define and detect "pure" authorship. As a result, institutions and platforms are left uncertain about how to evaluate and manage AI-influenced content, especially when it's been edited, paraphrased, or mixed with human input.

Because of the increase of usage of AI to generate text and the resulting lack of originality, substance to the text generated, alongside the widespread use of AI in today's work. A feasible approach is to use machine learning algorithms and deep learning algorithms, which can theoretically predict whether a given text is AI generated or not. In this work, we present multiple machine learning models, some of which are Random Forest, XGBoost, etc. These models are trained on datasets that contain both human and AI generated text. Our deep learning models like (DNN, BiLSTM, GRU,etc) allow for accurate prediction of AI generated text with high accuracy. In addition, we used pretrained models like Google's ELECTRA and RoBERTa, and AraBERTa, AraELECTRA on Arabic text for multilingual prediction. Preprocessing methods are applied to refine and optimize our data to ensure good model training.

To assess the increasing demand for reliable AI text detection, the work presented investigates and compares the performance of various machine learning and deep learning models. As part of our work, we compare traditional machine learning models (RF, XGboost, NB, Passive Aggressive) with deep learning architectures (DNN, BiLSTM, GRU, CNN), and an addition of pre-trained transformer models (ELECTRA, RoBERTa, AraBERTa, AraELECTRA) for detection in Arabic and English. This research strives to establish an effective framework for differentiating AI generated text from human writing, therefore, supporting the maintenance of ethical standards and authenticity throughout various communication environments.

II. Related Work

The study [2] put 14 popular AI detection tools to the test in real situations where a wrong call could seriously hurt a student's future. These tools were asked to spot normal student writing, content written entirely by AI, and AI writing that had been lightly tweaked. The results? Pretty surprising. Not one of the tools could reliably get it right 80% of the time. Nearly half of the AI-generated papers slipped through when students made just simple changes, and sometimes real student work got flagged by mistake. This is a serious issue that may result in unfair penalties or cheating being undetected; it's not just a technical one. It is obvious that schools require greater resources—and better approaches—to ensure that things remain equitable while AI writing continues to advance.

This study [3] looked into how well five popular AI detection tools could tell the difference between human-written engineering work and text generated by ChatGPT-3.5 and ChatGPT-4. While the tools did okay catching the older GPT-3.5 outputs, they really struggled with GPT-4 — and sometimes even flagged genuine student work as AI-generated by mistake. Each tool had its own personality: OpenAI's detector was quick to sound the alarm, sometimes raising flags a little too easily, while CrossPlag often missed AI-generated writing completely. These mixed results really show the bigger picture — current detection tools just aren't keeping pace with how fast AI is improving.

This study [4] took on a tricky challenge: figuring out when Arabic text is written by AI. Arabic's complicated diacritic marks make it even harder. So, the researchers used four powerful language models—AraELECTRA, AraBERT, XLM-R, and mBERT—and trained them on nearly 10,000 samples of both human and AI-written text, with and without those diacritics. The results were pretty amazing—the models spotted AI-generated text with 98.4% accuracy, way better than GPTZero's 62.7%. While the diacritics helped catch human writing, the best trick was actually removing them when testing. Even though this study focused on Arabic, the method could easily work for other languages too, giving teachers, editors, and publishers a big boost as AI writing gets smarter all the time.

This research [5] tackles a huge challenge: spotting when text is written by AI like ChatGPT-3.5, which sounds almost exactly like a real person. By using some of the newest pretrained AI models and training them on a mix of real and AI-written content, the system can now catch AI-generated text with an amazing 94% to 99% accuracy—way better than any human could. What this really shows is that even though AI writing is getting incredibly convincing, there are still smart tools that can almost always tell the difference. This study is a reminder of how far AI has come—and why we need equally clever ways to spot it if we want to keep trust and honesty alive online.

This paper [6] compares linguistic features that were handcrafted against a deep learning (RoBERTa) baseline for AI detection across two datasets (GPT-2, HC3). While a handcrafted feature based XGBoost achieved 94% F1 score on in-domain data, however, when applied to GPT-2 dataset, its performance dropped significantly ($20 - 22\%$ F1 score) which indicates poor generalization to different language models. On the other hand, RoBERTa achieved 98% and 63% respectively, which showcased adaptability. POS and dependency parsing analyses showed linguistic signatures of machine text, confirming the value of transformer-based methods for generalizable detection.

This article [7] introduces the CHEAT dataset—over 35,000 synthetic medical abstracts generated with ChatGPT—and pairs it with expert-written counterparts to test several machine learning pipelines. While basic classifiers distinguished AI and human abstracts with up to 92% accuracy, performance plunged when abstracts focused on similar medical subtopics, suggesting topic overlap can mask generation artifacts. The authors warn that in high-stakes domains like healthcare, misclassification risks (false negatives or false positives) could have serious consequences, calling for domain-aware detection strategies.

This study [8] probes a rich set of features—perplexity, semantic similarity, list lookup, document statistics, error-based cues, readability scores, AI feedback flags, and text embeddings—to distinguish human- from AI-generated (and AI-rephrased) prose across educational (English, French, German, Spanish) and news domains. By building the Multilingual Human-AI-Generated Text Corpus and training classifiers on both "from-scratch" and "rephrase" scenarios, the authors achieved detection rates that significantly outshine GPTZero and ZeroGPT baselines. Remarkably, their best rephrase detector lifted F1-score by 181.3% relative to GPTZero, demonstrating that multilingual, feature-based systems can robustly flag AI-tweaked content—though the continual evolution of prompting tactics.

This research [9] introduces a novel "BERTopic$\rightarrow$Gemini" pipeline to synthesize AI news articles that mirror the topics and style of 150 award-winning journalists' pieces. Training five classifiers on lexical, syntactic, and readability fingerprints, the Random Forest model led the pack with 98.3% accuracy, 0.984 precision, 0.983 recall, and an F1-score of 0.983. Feature-importance analyses and ANOVA pinpointed key stylistic markers—sentence-length range, paragraph-length coefficient of variation, verb ratio, complex sentence tags, and paragraph-length range—that most sharply separate human from machine t. By spotlighting these stylistic fingerprints, the study offers a high-fidelity guardrail against AI-driven disinformation in journalism and other critical domains.

This paper [10] addresses the limitations of AI text detectors in Arabic, a language undeserved by mainstream tools. After fine-tuning AraELECTRA and XLM-R on a dataset of 43K+ samples, alongside a custom dataset featuring ChatGPT and BARD outputs. The authors created a classifier model that outperforms GPTZero, OpenAI's detector, on the AIRABIC benchmark. An essential contribution was their dediacritization layer, which dealt with noise which was introduced by arabic diacritics, boosting accuracy from 81% to as much as 99 – 100%. The study highlights that language specific preprocessing and architecture tailoring can minimize detection gaps in Semitic languages.

This study [11] dives into AI-generated text detection in short dialectal Arabic which is not given enough attention despite its linguistic complexity. By merging two datasets (AIDialectGenerated, AIDialectRephrase) which cover five major dialects from these cities (Doha, Beirut, Cairo, Tunis, and Rabat), the author analyzes feature based and transformer based methods. AraBERT and AraELECTRA models outperform traditional baselines like (SVM, NB, Random Forest), achieving very good results (up to 98%) on the generated dataset and 83% on the paraphrased dataset. Interestingly, the latter was harder to detect due to its semantic closeness to text written by humans. The paper identifies syntactic and lexical cues such as (word length, readability) as crucial features,which highlight the importance of dialect-sensitive models and task-specific dataset design in Arabic NLP.

This work [12] offers a dataset of 10,000 samples (AI vs Human generated) and compares different detection methods, some include TF-IDF + Random Forest, and deep learning (CNN, LSTM). All these models got over 90% accuracy, the authors also analyzed explainability using SHAP (SHapley Additive exPlanations) and compared it against tools like GPTZero, OpenAI, Copyleaks, which revealed big shortcomings in the accuracy. Their study confirms that models that are lightweight with good features can offer deployability in real-world detection.

This paper [13] evaluates some numerical text representation techniques, some of these include Bag of Words, TF-IDF, fastText. Alongside sentence embeddings (MiniLM, distilBERT) to detect AI generated text in academic format. They used a balanced dataset (DAIGT V2), the authors did a lot of preprocessing, also, they applied classifier models like Logistic Regression, XGBoost, and LightGBM. In addition, they used TF-IDF tokenizer with LR and got the best accuracy of 99.82%, BoW and fast-Text also surpassed 96.5% across models. The work done to yield these results demonstrates well-defined/tuned models which outperform complex embeddings, lastly, these models are competitive for AI text detection.

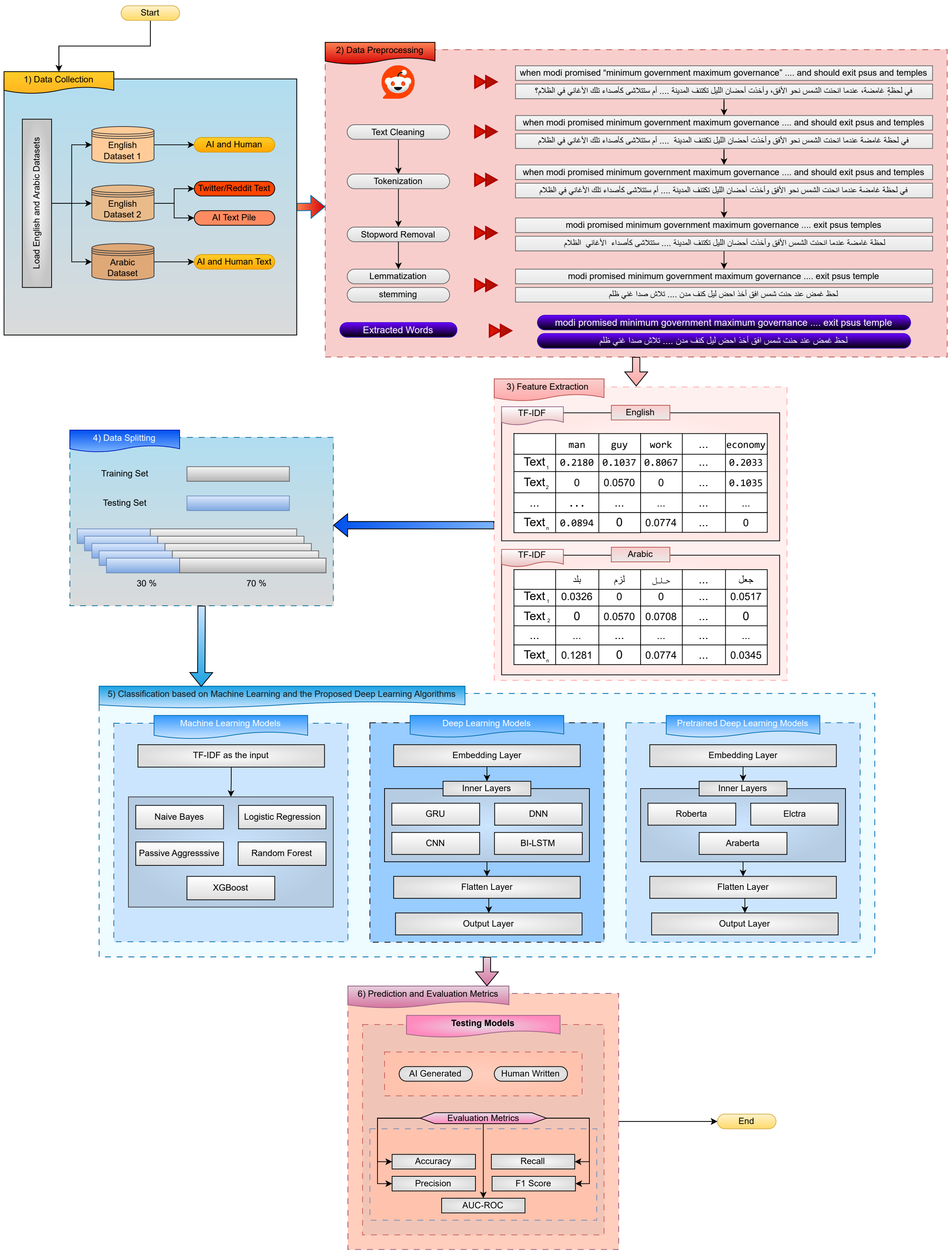
This study [14] proposes a classifier based on deep neural networks (DNNs) which utilizes a diverse set of linguistic and statistical features, such as token-level entropy, part-of-speech distributions, and n-gram surprisal scores, to distinguish AI-created and human-created writing. The authors relied dataset of 20,000 human-written as well as 20,000 news articles written by ChatGPT-3, and they extracted 150 features. They trained a five-layer 512 ReLU units multilayer perceptron (MLP). The DNN achieved 94.1%) accuracy on the unseen test set, that was better than both logistic-regression 87.3%) and SVM 89.5%) baselines, especially on small token sizes which was the most challenging to detect.

This study [15] focusses on AI-text identification as if it was an image classification issue by converting word-embedding sequences as grayscale images and making use of a custom convolutional neural network (CNN) framework. Each sentence is subjected to the steps of tokenization using FastText embedding with 300-dim vectors which converts text to images of dimensions 15 x 20 and input into a 6-layer CNN with residual connections. The model was trained on a corpus of academic abstracts 20k half ai and half human and achieved an 88.3% cross-domain detection which outperforms text-only models by 5%–7%. The authors also proposed a cyclic learning-rate scheduler ZigZag Scheduler in order to improve generalization for unseen data.

This study [16] implements a bidirectional gated recurrent unit (Bi-GRU) network on raw token embeddings to identify AI-generated spam reviews. Following standard preprocessing—lowercasing, punctuation removal, and word-level tokenization. Text is GloVe (200-dimensional) embedded and passed through a two-layer Bi-GRU with 256 units for each layer, and then through a dense classification layer. The dataset contains 5K genuine and 5K GPT-2 generated reviews. The model reached an accuracy of 96.2% and an F1-score of 0.95 which was way better than both uni-directional RNNs 89.4% and traditional ML models. Yet still, the model's classification was biased, or to be more direct, it was ultimately influenced by the number of input tokens.

This paper [17] presents the work *Detection of AI-generated Texts: A Bi-LSTM and Attention-Based Approach*, which was published in *IEEE Access* in April 2025. The authors proposed a new hybrid deep-learning framework that does the feature extraction as well as part-of-speech tagging and token-level processing before passing them to a Bidirectional LSTM with a self-attention layer. The model was trained and evaluated on the Kaggle DAGPap22 and Liyanage corpora. As a result, they were able to reach 94% with minimal processing overhead and fewer parameters than transformer-based models. Their plots showed nearly all features contributed to this high accuracy, and ablation studies proved the attention mechanism and POS features each added over 3% improvement to overall accuracy. This approach is beneficial when working with hardware constraints yet also needing an overall decent accuracy.

III. Proposed Methodology



Start
1) Data Collection
Load English and Arabic Datasets
English Dataset 1
AI and Human
English Dataset 2
Twitter/Reddit Text
AI Text Pile
Arabic Dataset
AI and Human Text
2) Data Preprocessing
Text Cleaning
Tokenization
Stopword Removal
Lemmatization
stemming
Extracted Words
when modi promised “minimum government maximum governance” .... and should exit psus and temples
في لحظةٍ غامضة، عندما انحنت الشمس نحو الأفق، وأخذت أحضان الليل تكتنف المدينة .... أم ستتلاشى كأصداء تلك الأغاني في الظلام؟
when modi promised minimum government maximum governance .... and should exit psus and temples
في لحظة غامضة عندما انحنت الشمس نحو الأفق وأخذت أحضان الليل تكتنف المدينة .... أم ستتلاشى كأصداء تلك الأغاني في الظلام
when modi promised minimum government maximum governance .... and should exit psus and temples
في لحظة غامضة عندما انحنت الشمس نحو الأفق وأخذت أحضان الليل تكتنف المدينة .... أم ستتلاشى كأصداء تلك الأغاني في الظلام
modi promised minimum government maximum governance .... exit psus temples
لحظة غامضة عندما انحنت الشمس الأفق وأخذت أحضان الليل تكتنف المدينة .... ستتلاشى كأصداء الأغاني الظلام
modi promised minimum government maximum governance .... exit psus temple
لحظ غمض عند حنت شمس افق أخذ احض ليل كنف مدن .... تلاش صدا غني ظلم
modi promised minimum government maximum governance .... exit psus temple
لحظ غمض عند حنت شمس افق أخذ احض ليل كنف مدن .... تلاش صدا غني ظلم
3) Feature Extraction
TF-IDF
English
man | guy | work | ... | economy
Text1 | 0.2180 | 0.1037 | 0.8067 | ... | 0.2033
Text2 | 0 | 0.0570 | 0 | ... | 0.1035
... | ... | ... | ... | ... | ...
Textn | 0.0894 | 0 | 0.0774 | ... | 0
TF-IDF
Arabic
بلد | لزم | حلل | ... | جعل
Text1 | 0.0326 | 0 | 0 | ... | 0.0517
Text2 | 0 | 0.0570 | 0.0708 | ... | 0
... | ... | ... | ... | ... | ...
Textn | 0.1281 | 0 | 0.0774 | ... | 0.0345
4) Data Splitting
Training Set
Testing Set
30 %
70 %
5) Classification based on Machine Learning and the Proposed Deep Learning Algorithms
Machine Learning Models
TF-IDF as the input
Naive Bayes
Logistic Regression
Passive Aggresssive
Random Forest
XGBoost
Deep Learning Models
Embedding Layer
Inner Layers
GRU
DNN
CNN
BI-LSTM
Flatten Layer
Output Layer
Pretrained Deep Learning Models
Embedding Layer
Inner Layers
Roberta
Elctra
Araberta
Flatten Layer
Output Layer
6) Prediction and Evaluation Metrics
Testing Models
AI Generated
Human Written
Evaluation Metrics
Accuracy
Recall
Precision
F1 Score
AUC-ROC
End

Fig. 1: Proposed Methodology

A. *Datasets Descriptions*

The first dataset used in this study has 487,235 entries and two columns (text and source). Each entry includes an essay that was either written by a human or generated by AI. The source column acts as the label, indicating whether the essay was written by a human (source = 0) or by AI (source = 1). The dataset size offers a solid basis for training and evaluating models for AI generated text detection. However, the dataset merely focuses on essays and formal texts, which could lead to poorer performance when given random, short or casual texts.

TABLE I: Features of the dataset

Feature	Type	Description
text	Categorical	Input text data
generated	Numerical	Label indicating source: 0 = human, 1 = AI

The second dataset, a balanced collection of short, casual texts written by both AI and humans, was created by carefully combining three different sources. The human-written content, which offers a broad range of common, conversational language, was sourced from two publicly available datasets of data collected from Reddit and Twitter comments. To maintain balance, we included an equal number of texts written by advanced AI models, selected from the AI Text Detection Pile dataset on Hugging Face. These samples reflect the style and tone commonly found in informal communication. Unlike the first dataset, which focuses on structured essays, this one brings in more variety and diversity, making it especially valuable for evaluating how well detection models perform in real-world, casual language scenarios.

TABLE II: Features of the third combined dataset (Sentiment Analysis + AI Detection)

Source Dataset	Feature	Type	Description
Twitter & Reddit	clean_comment / clean_text	Categorical	Cleaned user comments or tweets (all human-written) used for sentiment analysis
	category	Numerical	Sentiment label: -1 = Negative, 0 = Neutral, 1 = Positive
AI Detection	source	Categorical	Source name from the Pile dataset indicating the origin of the text
	id	Numerical	Unique identifier for each text entry
	text	Categorical	The text data used for detecting whether it is AI-generated or human-written

For the third and final dataset, we're working with Arabic text entries split into training and test groups—about 6,700 for training and 1,450 for testing. The actual Arabic text, and a label showing whether it was written by a human (0) or created by AI (1). This dataset is Pretty good for seeing how well models can tell the difference between human and AI-generated Arabic writing.

TABLE III: Features of dataset-arabic-generated-text

Dataset	Feature	Type	Description
Train	text_id	Numerical	Unique identifier for each text entry
	text	Categorical	Input text data (in Arabic)
	generated	Numerical	Label indicating source: 0 = human-written, 1 = AI-generated
Test	text_id	Numerical	Unique identifier for each text entry
	text	Categorical	Input text data (in Arabic)
	generated	Numerical	Label indicating source: 0 = human-written, 1 = AI-generated

B. *Data Preprocessing*

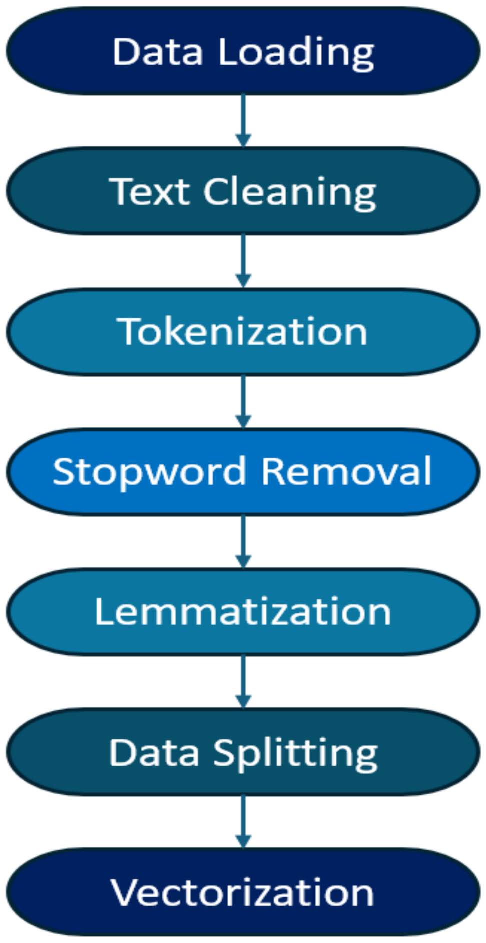


Fig. 2: Preprocessing Steps

One of the core concepts of data science is data preparation, which is necessary to prepare datasets for models to use and evaluate. During this crucial stage, a number of actions are taken to improve the quality of the data.

1) Dataset 1: Human vs AI Text:

1) **Data Loading:**
   We started by loading the AI_Human.csv file using the kagglehub library. This file includes many text samples, and each one shows whether it was written by a human or by an AI.

2) **Text Cleaning:**
   To make the text easier to work with, we used a function to clean it. We removed any Special or strange characters, extra spaces, and changed all the letters to lowercase. This helped us keep everything consistent.

3) **Tokenization:**
   After cleaning, we split each sentence into individual words using a tool called word_tokenize from the NLTK library.

4) **Stopword Removal:**
   We removed common words like “the”, “is”, and “and” using NLTK’s list of English stopwords. These words don’t add much meaning, so it’s better to take them out.

5) **Lemmatization:**
   Then, we used a tool called WordNetLemmatizer to reduce words to their original form. For example, “running” becomes “run”. This makes the text easier to analyze.

6) **Data Splitting:**
   The dataset was split into training and testing sets to evaluate model performance effectively. We used a standard 70/30 ratio, with 70% of the data used for training and the remaining 30% reserved for testing.

7) **Vectorization:**
   After splitting, we formatted it so the model could understand it. For machine learning models, we converted the words into numerical counts using `CountVectorizer()`. Then, `TfidfTransformer()` was then used to scale these counts according to the relative importance of each word in the entire dataset. For deep learning models, we used `Tokenizer()` from Keras to convert the text into sequences of numbers.

2) Dataset 2: Twitter, Reddit, and AI Text:

1) **Data Loading:**
   We loaded text samples from Twitter and Reddit using kagglehub library. We also used the “ai-text-detection-pile” dataset from Hugging Face, which has texts labeled as human or AI-generated.

2) **Preparing Reddit and Twitter DataFrames:**
   We ensured that both datasets had consistent column names and removed any unnecessary columns. Specifically, we renamed the text columns in the Reddit and Twitter DataFrames to a unified name, "text", and dropped the "category" columns, as they were not needed for the source classification task. The Twitter and Reddit DataFrames were then concatenated into a single DataFrame called df1. A new column named "source" was added with all values set to 0, indicating that these texts were human-written.

3) **Removing Duplicates and Missing Values:**
   We cleaned the data by removing any duplicate or empty rows to ensure data quality.

4) **Preparing the AI-Text-Detection-Pile DataFrame:**
   We filtered the Hugging Face dataset to include only AI-generated texts. To maintain balance, we selected a number of AI texts equal to the number of human texts. The "id" column was dropped, and we retained only rows with a "source" value of 1, indicating AI-generated content.

5) **Combining Human and AI Texts:**
   We joined the human and AI datasets into one final dataset and shuffled the rows so everything was mixed up.

6) **Text Cleaning:**
   To make the text easier to work with, we used a function to clean it. We removed any Special or strange characters, extra spaces, and changed all the letters to lowercase. This helped us keep everything consistent.

7) **Tokenization:**
   The cleaned text was then broken down into individual words using the word_tokenize function from the NLTK library.

8) **Stopword Removal:**
   We removed common English words to reduce noise.

9) **Lemmatization:**
   Then we used NLTK’s WordNetLemmatizer to reduce each word to its base or dictionary form. This helps group together different forms of the same word and simplifies the data for analysis.

10) **Data Splitting:**
    The dataset was split into training and testing sets to evaluate model performance effectively. We used a standard 70/30 ratio, with 70% of the data used for training and the remaining 30% reserved for testing.

11) **Vectorization:**
After splitting, we formatted it so the model could understand it. For machine learning models, we converted the words into numerical counts using `CountVectorizer()`. Then, `TfidfTransformer()` was then used to scale these counts according to the relative importance of each word in the entire dataset. For deep learning models, we used `Tokenizer()` from Keras to convert the text into into sequences of numbers.

3) Dataset 3: Arabic Human vs AI Text:

1) **Data Loading:**
We loaded Arabic text samples from kagglehub. These samples were in two files—train_set.csv and dev_set.csv—and they were labeled to show if they were written by a human or AI.

2) **Combining the Data:**
We joined the two datasets into one so we could work with all the data at once.

3) **Preparing the DataFrame:**
We removed the text_id column as it's not needed for the text analysis, and we renamed the generated column to source for clarity and consistency, indicating the text's origin (human or AI).

4) **Removing Duplicates:**
Duplicate entries were identified and removed to ensure data integrity.

5) **Text Cleaning:**
To make the text easier to work with, we used a function to clean it. We removed any Special or strange characters, and extra spaces. This helped us keep everything consistent.

6) **Tokenization:**
The cleaned text was then broken down into individual words using the word_tokenize function from the NLTK library.

7) **Stopword Removal:**
We removed common stopwords using NLTK's list of Arabic stopwords. These words don't add much meaning, so it's better to take them out.

8) **Stemming:**
We then applied stemming to the Arabic text to reduce words to their root forms. This step helped us group together words that shared the same base, even if they appeared in different forms. It simplified the vocabulary and made it easier for our model to focus on the core meaning of each word rather than getting distracted by slight variations.

9) **Data Splitting:**
The dataset was split into training and testing sets to evaluate model performance effectively. We used a standard 70/30 ratio, with 70% of the data used for training and the remaining 30% reserved for testing.

10) **Vectorization:**
After splitting, we formatted it so the model could understand it. For machine learning models, we converted the words into numerical counts using `CountVectorizer()`. Then, `TfidfTransformer()` was then used to scale these counts according to the relative importance of each word in the entire dataset. For deep learning models, we used `Tokenizer()` from Keras to convert the text into into sequences of numbers.

C. *Data Visualization*

We relied on data visualization throughout our project to better understand the large and complex text datasets we were working with. Instead of just looking at raw numbers or blocks of text, using visual tools helped us spot patterns, trends, and unusual cases much more easily. We examined differences in frequency of words and text length between sources, for example, or the proportion of AI-generated material vs human-written text. These realizations not only helped us develop and refine our models, but they also greatly facilitated the clear and simple explanation of our results.

1) Dataset 1 Visualization:

1) **Visualizing Class Distribution in the Dataset:**

Before training our model, we took a moment to look at how the data was divided between human-written and AI-generated texts. The dataset comprised, 60% human-authored and 40% AI-generated texts. Though not evenly distributed, the ratio fell within acceptable bounds, so we retained the original split.

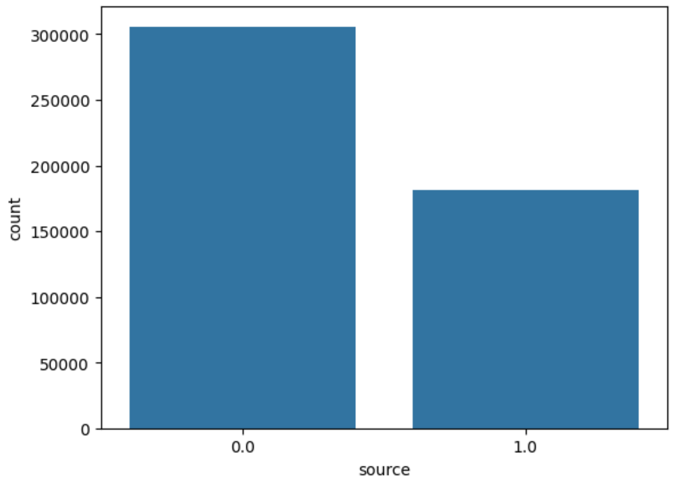


Fig. 3: Dataset 1 Class Distribution

2) **Analysis Of Text Length Distribution:**

To get a better feel for the type of content we were working with, we looked at the distribution of text lengths across the dataset. We plotted how long each text sample was, just to see whether we were mostly dealing with short comments or longer pieces. The results showed that the majority of the entries were quite lengthy, which made sense since the dataset is made up of essays. This confirmed that our data was more suited for analyzing structured, long-form writing rather than casual or short texts.

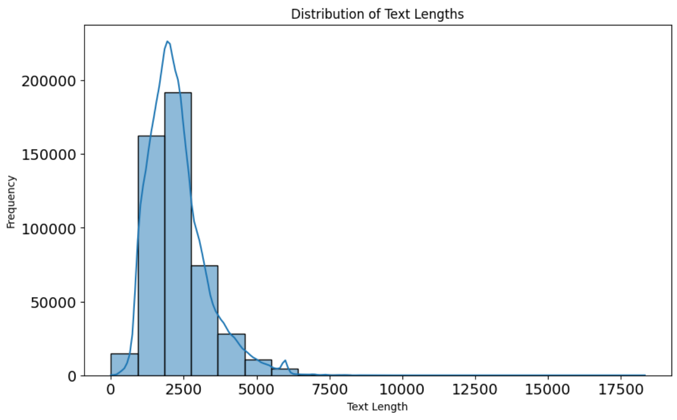


Fig. 4: Dataset 1 Text Length Distribution

3) **Exploring the most Common Words in the Dataset:**

We started by checking which words showed up most often across all the texts—both human and AI-written. No big surprises there: words like 'the,' 'and,' and 'of' topped the list for both. While this told us the language structure looked natural, it also meant these super common words weren't giving us any real insights. So in our next pass, we stripped out all the basic stopwords. That way, we could zero in on the more interesting words—the ones that actually help spot whether a human or an AI wrote something.

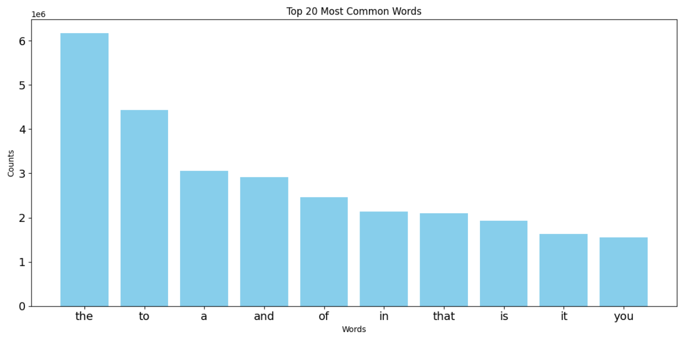


Fig. 5: Dataset 1 Most common Text By Human

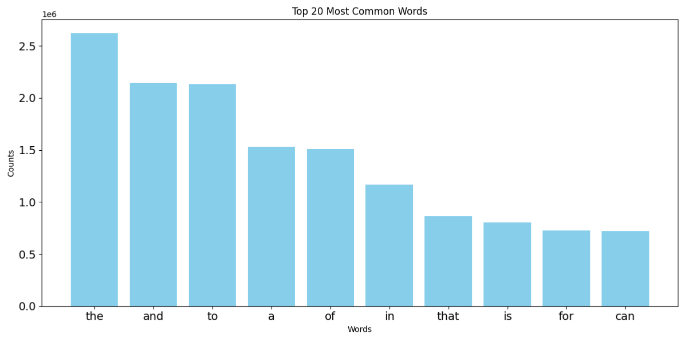


Fig. 6: Dataset 1 Most common Text By AI

4) **Exploring the most Common Words after removing the stopwords:**

After filtering out common stopwords from both human and AI-generated texts, we re-examined the most frequently used words. This time, the results were much more insightful—words with real meaning and context began to stand out. Instead of generic fillers like "the" or "and," we saw more content-specific terms that reflected the actual topics and writing styles in the dataset. This step helped us get a clearer sense of the language patterns used by each source.

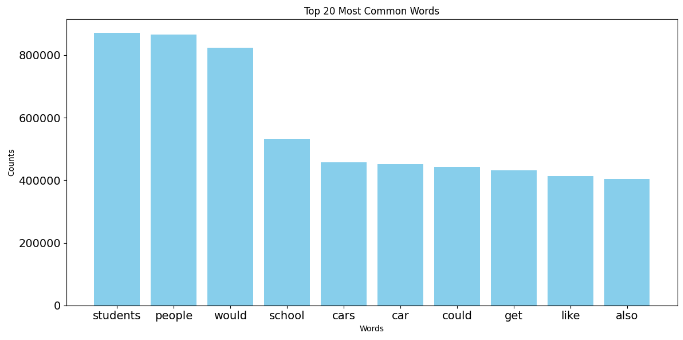


Fig. 7: Dataset 1 Most common Text By Human after Stopwords Removal

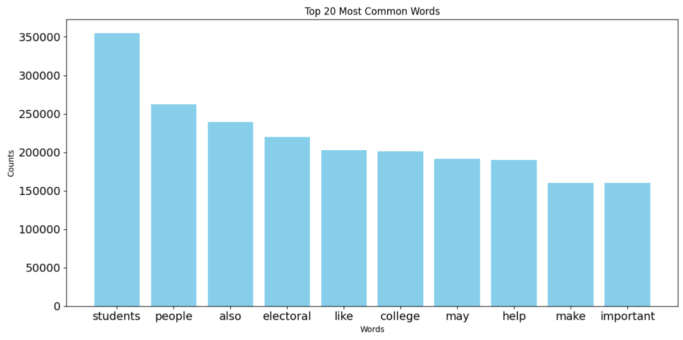


Fig. 8: Dataset 1 Most common Text By AI after Stopwords Removal

5) **Exploring the most Common Words after performing lemmatization:**

To refine our analysis further, we applied lemmatization to the text, reducing words to their root forms. This helped group together variations of the same word. After plotting the most frequent lemmatized words, we noticed a clearer and more unified vocabulary distribution. This gave us a more accurate understanding of common themes and language usage in both human and AI-generated content.

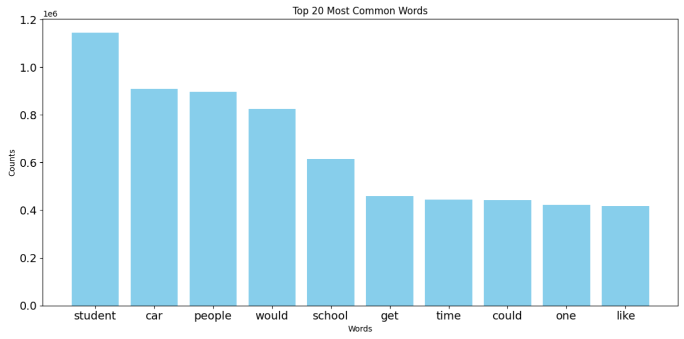


Fig. 9: Dataset 1 Most common Text By Human after Lemmatization

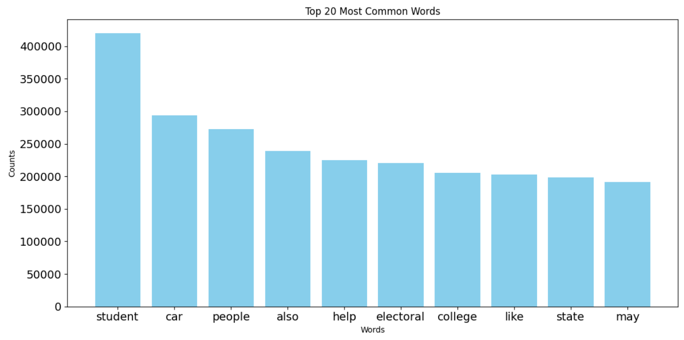


Fig. 10: Dataset 1 Most common Text By AI after Lemmatization

2) Dataset 2 Visualization:

1) **Visualizing Class Distribution in the Dataset:**

In the second dataset, we verified that the texts are distributed evenly, 50% human-authored and 50% AI-generated. Because this perfectly balanced split eliminates any class-imbalance concerns, we retained the dataset's original structure for training.

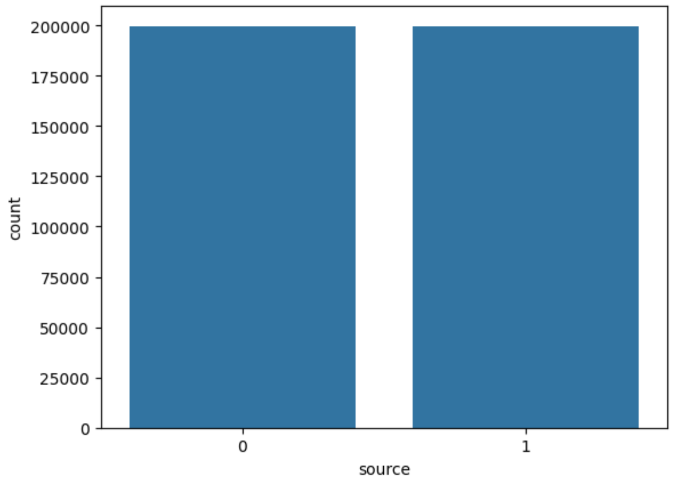


Fig. 11: Dataset 2 Class Distribution

2) **Analysis Of Text Length Distribution:**

For the second dataset, which consists entirely of shorter sentences rather than full essays, we performed the same length-distribution analysis. As expected, the text samples were substantially shorter on average—most entries fell into the single-sentence range. This confirmed that the dataset's scope is focused on concise, sentence-level writing rather than extended, structured documents.

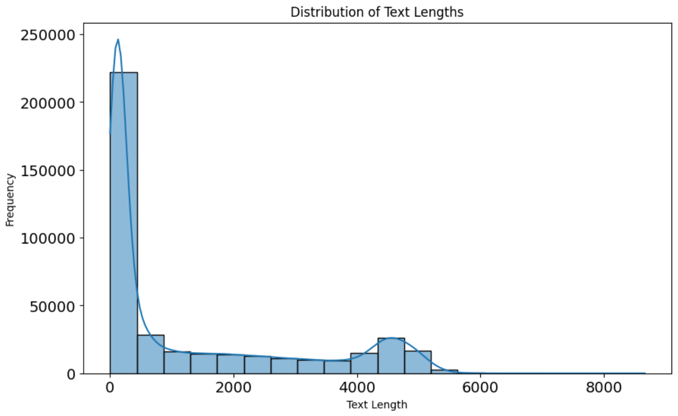


Fig. 12: Dataset 2 Text Length Distribution

3) **Exploring the most Common Words in the Dataset:**

We followed the same approach—identifying the most frequently occurring words across both human- and AI-written sentences. Similar to the first dataset, common function words like "the," "to," and "a" dominated the initial results. This reinforced that the language patterns were consistent and natural. As before, we removed standard stopwords to better highlight more meaningful vocabulary that could contribute to distinguishing between human and AI authorship. The results aligned closely with those from the first dataset, confirming a similar linguistic structure.

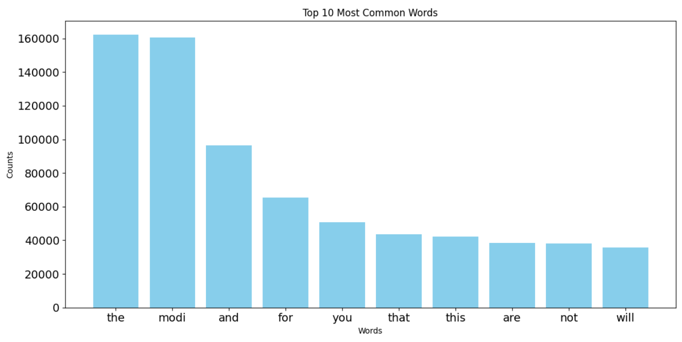


Fig. 13: Dataset 2 Most common Text By Human

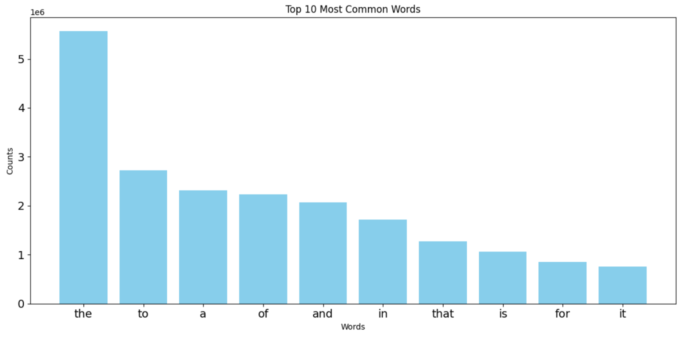


Fig. 14: Dataset 2 Most common Text By AI

4) **Exploring the most Common Words after removing the stopwords:**

Similarly, once we filtered out the common stopwords, we observed a clearer and more meaningful set of frequently used words. Like the first dataset, content-specific terms emerged, providing valuable insights into the topics and stylistic differences between human and AI-generated sentences. This step again proved effective in highlighting distinct language patterns within the dataset.

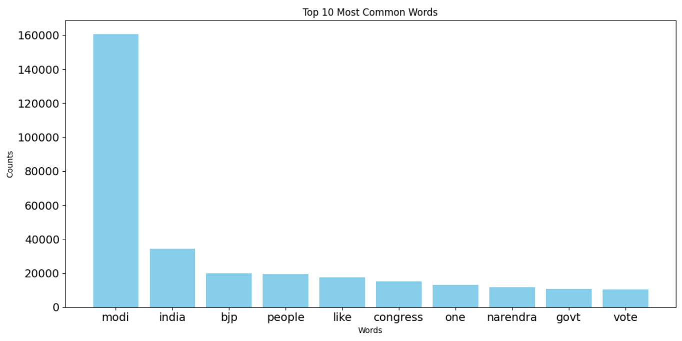


Fig. 15: Dataset 2 Most common Text By Human after Stopwords Removal

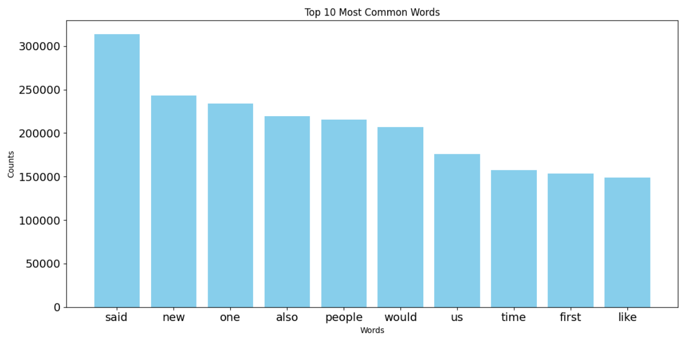


Fig. 16: Dataset 2 Most common Text By AI after Stopwords Removal

5) **Exploring the most Common Words after performing lemmatization:**

We applied the same lemmatization process—reducing each word to its base form before recalculating frequencies. As with the essay-length texts, this normalization brought together different inflections and revealed a more cohesive set of root terms. The most frequent lemmatized words now painted an even clearer picture of the core vocabulary across human- and AI-generated sentences, confirming that lemmatization consistently enhances our ability to detect thematic and stylistic patterns, regardless of text length.

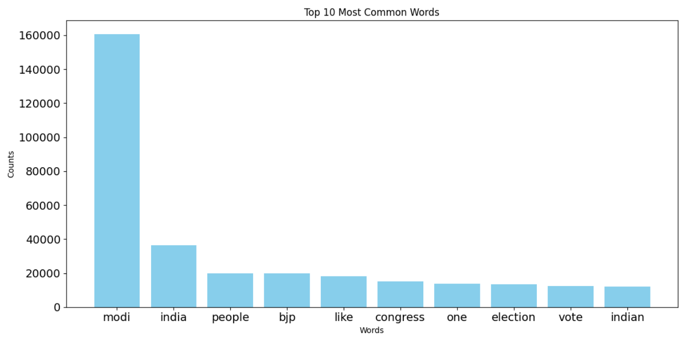


Fig. 17: Dataset 2 Most common Text By Human after Lemmatization

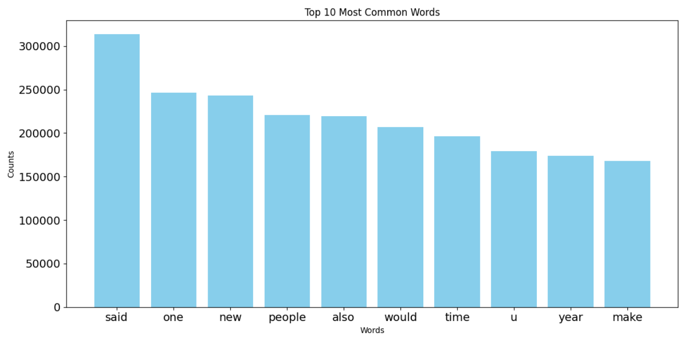


Fig. 18: Dataset 2 Most common Text By AI after Lemmatization

1) **Visualizing Class Distribution in the Dataset:**

3) Dataset 3 Visualization:
To better understand the third dataset, we started by plotting the distribution of human and AI-generated Arabic texts. Compared to the previous datasets, this one was noticeably smaller in size, but still offered valuable insight. The class distribution was nearly even, with a slight difference that wasn't significant enough to raise concerns. This early check helped us ensure a fair starting point for our analysis, especially given the language difference and the smaller scale of the data.

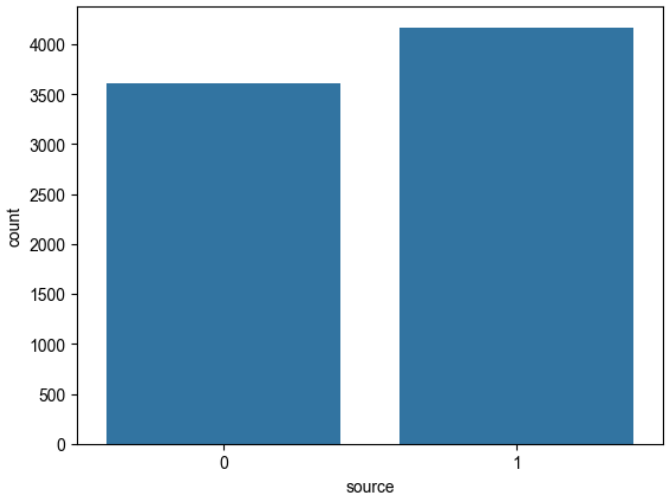


Fig. 19: Dataset 3 Class Distribution

2) **Analysis Of Text Length Distribution:**

To get a sense of how long the Arabic texts were in our third dataset, we plotted their length distribution. Unlike the first dataset, which contained longer essays, this one resembled the second English dataset with much shorter entries. Most of the texts were brief, often just a few words or sentences. This insight was important, as it helped us set realistic expectations for how much information each entry could provide and guided how we approached preprocessing and model training.

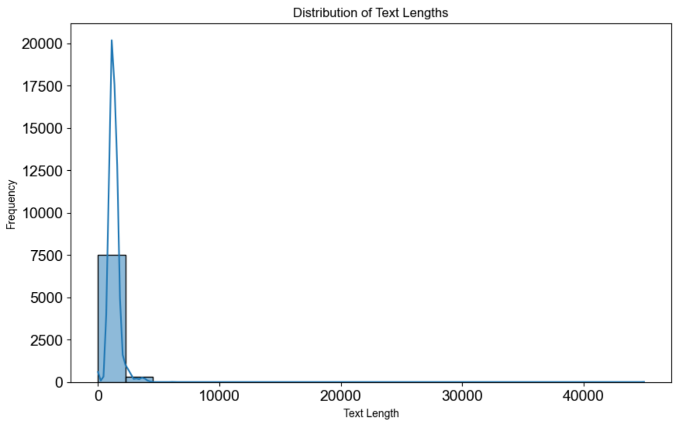


Fig. 20: Dataset 3 Text Length Distribution

3) **Exploring the most common words in the dataset:**

We also visualized the most frequently used words in both the human and AI-generated texts from the Arabic dataset. As expected, the top words were mostly common stopwords—words that appear often but carry little meaningful information, like conjunctions and prepositions. This reinforced the need to remove these stopwords during preprocessing to focus on the more informative parts of the text that could help our model better distinguish between human and AI content.

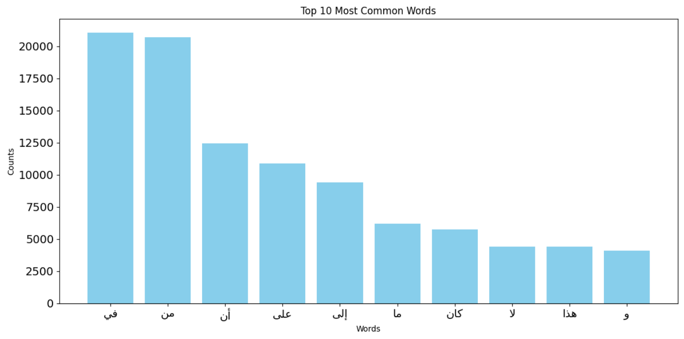


Fig. 21: Dataset 3 Most common Text By Human

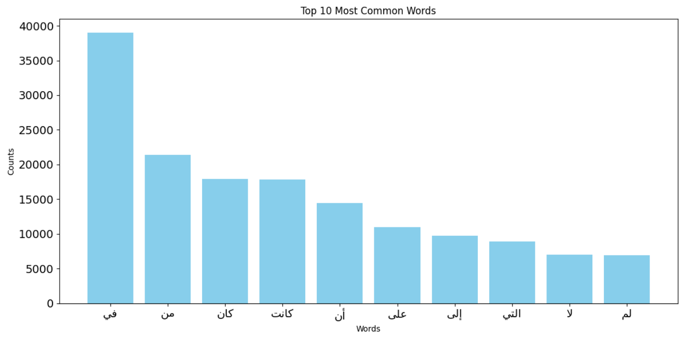


Fig. 22: Dataset 3 Most common Text By AI

4) **Exploring the most common words after stopwords removal:**

Similarly, once we filtered out the common stopwords, we observed a clearer and more meaningful set of frequently used words. Like the first two datasets, content-specific terms emerged, providing valuable insights into the topics and stylistic differences between human and AI-generated sentences.

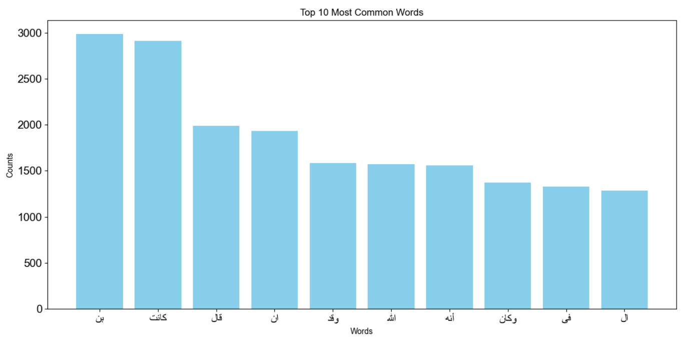


Fig. 23: Dataset 3 Most common Text By Human after Stopwords Removal

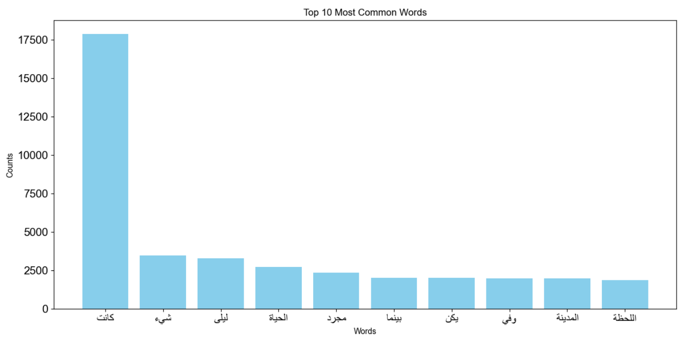


Fig. 24: Dataset 3 Most common Text By AI after Stopwords Removal

5) **Exploring the most common words after stemming:**

After applying stemming to the Arabic text, we plotted the most frequently used words once again. This step trimmed the words down to their core forms, which helped reduce variation caused by different word endings. As a result, the plot became even more focused, highlighting the essential vocabulary used across both human and AI texts. This made patterns in word usage clearer and brought us closer to understanding the underlying structure of the language in our dataset.

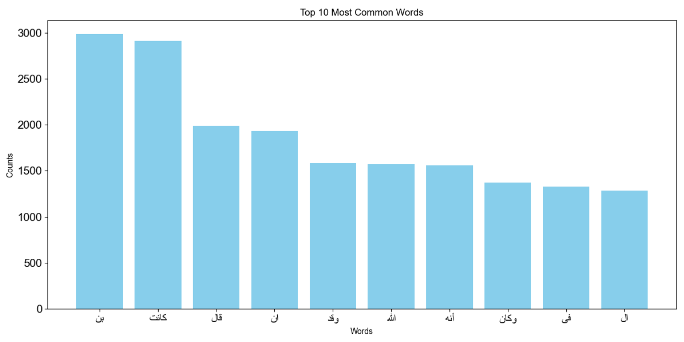


Fig. 25: Dataset 3 Most common Text By Human after Stemming

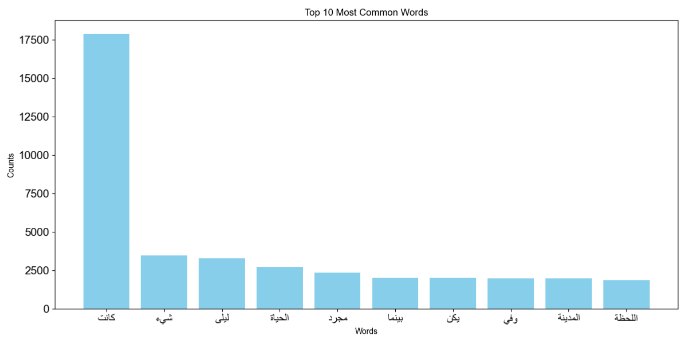


Fig. 26: Dataset 3 Most common Text By AI after Stemming

D. *Used Algorithms*

The three datasets were used to train five machine learning models: Logistic Regression, Random Forest, Naive Bayes, Passive Aggressive, XGBoost. Additionally, four deep learning models were employed: Deep Neural Network, Bidirectional Long Short-Term Memory, (BiLSTM), Convulotional Neural Network (CNN), A Gated Recurrent Unit (GRU). Furthermore, Google's pre-trained models, RoBERTa, ELECTRA, and their Arabic equivalents, AraBERTa, AraELECTRA were utilized and finetuned to the dataset to enhance their performance. Detailed performance results, visualizations, and discussions of these models are provided later in this paper.

1) **Bi-LSTM**
A Bidirectional Long Short-Term Memory (Bi-LSTM) is a derived version of LSTM model, which is designed to process sequential data in both directions (forward, backward) simultaneously. Although, LSTM processes a sequence from start to finish, the Bi-LSTM consists of two layers of LSTM: one processes the same input in forward direction (past to future), the other does the same but in reverse direction (future to past). Moreover, the outputs from the two directions are combined, usually by averaging, dependencies and concentration at each time step. Therefore, this allows the Bi-LSTM to provide a more comprehensive understanding of the entire context. Lastly, this Bi-LSTM is powerful for multiple NLP tasks.

2) **DNN:**
The Deep Neural Network model is an artificial intelligence model that aims to mimic the human brain in the way it processes information. This network is distinctively known for its hidden layers between the input and output layers. Unlike simple neural networks, the "deep" its name highlights the increased number of layers. Therefore, allowing for DNNs to learn complex patterns and relationships inside the given data. Each one of the DNNs layers allows for the completion of highly complex mathematical transformations on the input received, steadily and progressively extracting

more refined features as the data is moved through the network. This, in hindsight, enables DNNs to tackle challenging tasks like NLP, image recognition and speech recognition with very high accuracy. Overall, DNNs serve as a powerful and effective model for AI text detection.

3) **GRU:**
A GRU (Gated Recurrent Unit) is a type of RNN architecture which is made to effectively process sequential data, for example, text or speech. Initially, it was introduced as an alternative to LSTM. The two of them aim to tackle the vanishing gradient problem which is common in traditional RNNs. The innovation of this model is in the "gating mechanisms". These gating mechanisms are made up of the reset and update gate. The reset gate decides decides how much of the hidden state previously determined, should be forgotten, which in turn, allows for the model to discard irrelevant past stored information. The update gate determines how much of the prior hidden state should be kept and how much of a new hidden state should be included. With the process of selective allowance of data through these gates, GRU can effectively learn and remember sequences, hence, proving to be well suited for task like NLP.

4) **CNN:**
Although generally known for image processing, CNNs work well for text classification by converting text into 1D image. The main idea is that kernels move across numerically represented word embeddings in a sentence, then it identifies local patters. Moreover, different filters can capture different linguistic features like collocations, phrases. After the extraction of such features, max pooling layers are generally applied to down-sample feature maps and select the most notable features. Lastly, these features are flattened into fully connected layers, that are then used to classify into predefined categories, similar to image classification. This model allows for effective hierarchical representations of text and extract important information for tasks like ours, which is text classification.

5) **RoBERTa/AraBERT:**
Roberta is a 12-layer Transformer encoder. Roberta masks the input in order to eliminate redundant words by masking them additionally it introduces a subtle randomness in the masking that can be controlled by the temperature parameter to prevent the model from overfitting. After eliminating the redundant data in the text, the model proceeds to analyze the depth of the context in the written text. This process is helpful for noticing twists and tone changes in the input text. If the text follows a stable or too safe almost ideal yet repetitive pattern it increases its chances to likely be classified as ai generated. However, if the given text has a decent variability of tones or even flaws in the way that the context is presented after masking the text, it increases the chances that this is likely something that only a human can write. The model uses attention mechanisms to give appropriate weight to the presented words not just process word for word in the text but to have a better understanding of the overall context and how it's presented.

6) **ELECTRA/AraELECTRA:**
Roberta is a 12-layer Transformer encoder. Roberta masks the input in order to eliminate redundant words by masking them additionally it introduces a subtle randomness in the masking that can be controlled by the temperature parameter to prevent the model from overfitting. After eliminating the redundant data in the text, the model proceeds to analyze the depth of the context in the written text. This process is helpful for noticing twists and tone changes in the input text. If the text follows a stable or too safe almost ideal yet repetitive pattern it increases its chances to likely be classified as ai generated. However, if the given text has a decent variability of tones or even flaws in the way that the context is presented after masking the text, it increases the chances that this is likely something that only a human can write. The model uses attention mechanisms to give appropriate weight to the presented words not just process word for word in the text but to have a better understanding of the overall context and how it's presented.

7) **Random Forest:**
A random forest is a machine learning method which work by constructing multiple decision trees during training and then outputs the class that is mode of these classes, which is the classification. Furthermore, each tree in the forest is build using bootstrapping (training on subset of data) and only a random subset of features is considered, thus, helping the de-correlation of the trees. Lastly, the model is very robust and less prone to overfitting than a single decision tree and still maintains high accuracy and handling of relationships as well.

$$\hat{y} = \operatorname{mode}\{T_b(x)\}_{b=1}^{B} \tag{1}$$

8) **logistic Regression:**
A basic statistical model called logistic regression used primarily for binary classification, however, it can be extended for multi-class problems. The model fits the data to a logistic function (sigmoid) to model the probability of binary outcome. The sigmoid function places any real-world values into a threshold of between 0 and 1. In addition, the model uses numerical represen-

tations of text (TF-IDF, etc) as input. Lastly, whilst it is fundamentally a linear model, its ability to provide probability estimates makes it an effective and widely used classifier.

$$P(y = 1 \mid x) = \frac{1}{1 + e^{-(w^T x + b)}} \tag{2}$$

9) **Naive Bayes:**
Naive Bayes is a leading and one of the most known, effective ways for text classification. The model calculates the probability of a feature belonging to a class (e.g, "AI", "Human") based on the existence of frequency of specific words within that feature. Moreover, the presence of a word in a document is independent of the presence of other words in a class, this is the "naive" assumption. Given the equation:

$$\hat{y} = \arg\max_{C_k} \left[\log P(C_k) + \sum_{i=1}^{n} \log P(x_i \mid C_k) \right] \tag{3}$$

The model would calculate the probabilities in theory like this: "What is the probability that someone will not play Football given that the weather is Rainy, Hot, High humidity, and No wind?" This model, despite its simplistic approach, is one of the most accurate models in text classification.

10) **XGBoost:**
XGBoost is a gradient boosting library designed to be highly efficient and optimized. In addition, it is an implementation of gradient boosting, which is an ensemble technique, thus, correcting errors made by previous trees by sequentially placing new trees. Moreover, the model is distinguished by its regularization techniques to prevent overfitting. Some of the optimizations that are done include: handling missing values, support parallel processing, and a tree pruning algorithm.

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) \tag{4}$$

11) **Passive Aggressive:**
The Passive Aggressive classifier is an online learning algorithm that is widely used for classification tasks. The main idea is that it can update its weights as new data is coming in. Thus, it's very well-suited for learning in large scales. In addition, the "passive" part of the name means that if a prediction is correct, the model remains unchained. On the other hand, if the prediction is incorrect, it represents the "aggressive" in the name. Moreover, the model is highly responsive to new data, making it effective for tasks like our work in AI text detection.

$$\text{If } \hat{y}_t \neq y_t : \quad w_{t+1} = w_t + \tau_t y_t x_t \tag{5}$$

$$\text{If } \tau_t = \frac{\max\left(0, 1 - y_t w_t^T x_t\right)}{\|x_t\|^2} \tag{6}$$

E. *Performance Metrics*

A variety of criteria are used to assess classification models to determine their accuracy and dependability. Precision indicates that for each class, which of its predicted positives were correct. Moreover, the recall measures how much of the true classes the model predicted correctly. In addition, the F1 score is used to see the harmonic mean of precision and recall, combining it into a single value where only when both are high, the score is high. Accuracy is simple and reports the fraction of all predictions that were right. Macro averaged metric is the unweighted mean of a metric along classes, each class being treated equally. Additionally, the weighted average metrics reflects overall effectiveness even when classes are imbalanced. Lastly, support is the count of true samples per class.

$$\begin{aligned}
\text{Precision}_c &= \frac{TP_c}{TP_c + FP_c}, \\
\text{Recall}_c &= \frac{TP_c}{TP_c + FN_c}, \\
F1_c &= 2 \cdot \frac{\text{Precision}_c \times \text{Recall}_c}{\text{Precision}_c + \text{Recall}_c}, \\
\text{Accuracy} &= \frac{\sum_{c=1}^{C} TP_c}{N}, \\
\text{Macro Avg Metric} &= \frac{1}{C} \sum_{c=1}^{C} \text{Metric}_c, \\
\text{Weighted Avg Metric} &= \sum_{c=1}^{C} \frac{N_c}{N} \text{Metric}_c, \\
\text{Support}_c &= N_c,
\end{aligned}$$

IV. Results and Analysis

The results collected our machine learning models (Random Forest, Logistic Regression, Naive Bayes, XGBoost) are shown below. Alongside the Deep Learning models used (DNN, BiLSTM, CNN, GRU) followed by transformer models (RoBERTa, ELECTRA, AraBERT, AraELECTRA).

A. *Machine Learning*

The following results are from the first dataset.

TABLE IV: Performance comparison across ML classifiers

Metric	RF	LR	NB	XGBoost	PA
Precision	1.00	0.99	0.95	0.99	1.00
Recall	1.00	0.99	0.95	0.99	1.00
F1-Score	1.00	0.99	0.95	0.99	1.00
Support	146,171	146,171	146,171	146,171	146,171
Model Accuracy	0.9978	0.9921	0.9492	0.9944	0.9982

The following results are from the second dataset.

TABLE V: Performance comparison across ML classifiers

Metric	RF	LR	NB	XGBoost	PA
Precision	0.98	0.99	0.96	0.98	0.99
Recall	0.98	0.99	0.96	0.98	0.99
F1-Score	0.98	0.99	0.95	0.98	0.99
Support	119,782	119,782	119,782	119,782	119,782
Model Accuracy	0.9780	0.9873	0.9551	0.9816	0.9877

The following results are from the third dataset.

TABLE VI: Performance comparison across ML classifiers

Metric	RF	LR	NB	XGBoost	PA
Precision	0.98	0.99	0.96	0.98	1.00
Recall	0.97	0.99	0.96	0.98	1.00
F1-Score	0.97	0.99	0.96	0.98	1.00
Support	2 336	2 336	2 336	2 336	2 336
Model Accuracy	0.9739	0.9910	0.9576	0.9846	0.9953

B. *Deep Learning*

The following results are from the first dataset.

TABLE VII: Comparison of Deep Learning Model Evaluation Metrics

Metric	CNN	Bi-LSTM	GRU	DNN
Accuracy	0.9991	0.9984	0.9979	0.9976
Precision	0.9991	0.9984	0.9979	0.9976
Recall	0.9991	0.9984	0.9979	0.9976
F1-Score	0.9991	0.9984	0.9979	0.9976
AUC-ROC	1.0000	0.9999	0.9999	0.9999

The following results are from the second dataset.

TABLE VIII: Comparison of Deep Learning Model Evaluation Metrics

Metric	CNN	Bi-LSTM	GRU	DNN
Accuracy	0.9910	0.9924	0.9921	0.9903
Precision	0.9910	0.9925	0.9921	0.9903
Recall	0.9910	0.9924	0.9921	0.9903
F1-Score	0.9910	0.9924	0.9921	0.9903
AUC-ROC	0.9992	0.9996	0.9996	0.9995

The following results are from the third dataset.

TABLE IX: Comparison of Deep Learning Model Evaluation Metrics

Metric	Bi-LSTM	CNN	GRU	DNN
Accuracy	0.9672	0.9846	0.9750	0.9917
Precision	0.9682	0.9846	0.9750	0.9917
Recall	0.9672	0.9846	0.9750	0.9917
F1-Score	0.9672	0.9846	0.9750	0.9917
AUC-ROC	0.9947	0.9978	0.9959	0.9985

In analyzing the model accuracies on the three datasets, we observed varying performance across different performance metrics for the nine models evaluated. All ML models achieved an accuracy of over 95%, with the highest of them in the three datasets being XGBoost and PA. On the other hand, the deep learning models proved to be better and higher in accuracy. The average minimum accuracy is 98% among the four deep learning models. Regarding the transformer models, all of them had an accuracy over 98%, and some even exceeded 99%. Thus, this proves that deep learning models are better at learning complex patterns present in our datasets.

The upcoming plots will illustrate the confusion matrix and the ROC curve. These graphs were generated by each model.
The following results are from the first dataset.

a) **CNN**

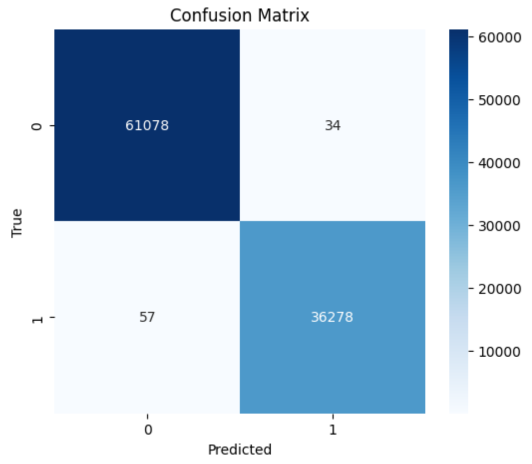


Fig. 27: First dataset confusion matrix

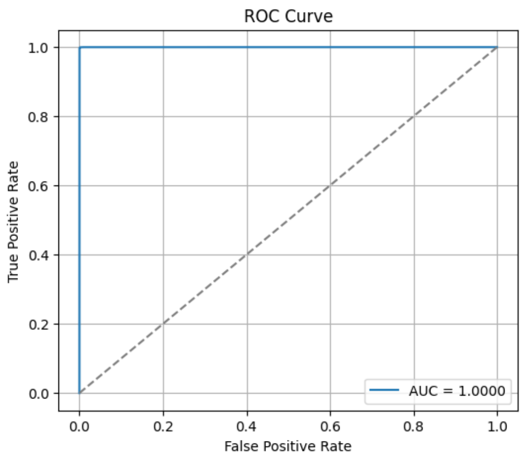


Fig. 28: First dataset ROC curve

b) **BiLSTM**

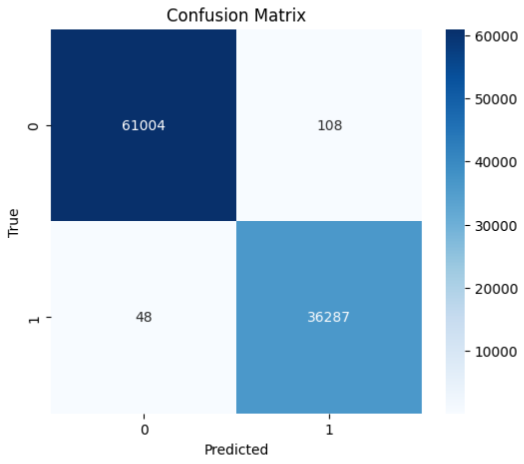


Fig. 29: First dataset confusion matrix

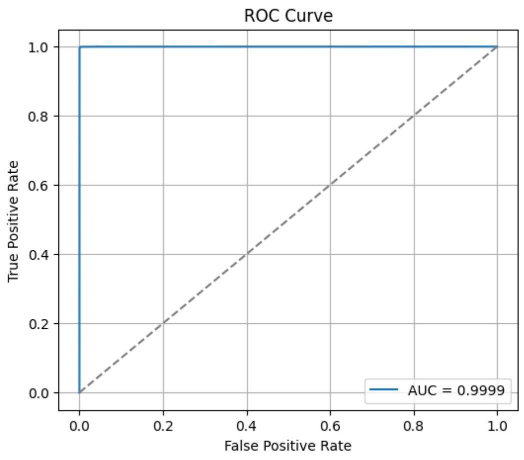


Fig. 30: First dataset ROC curve

c) **GRU**

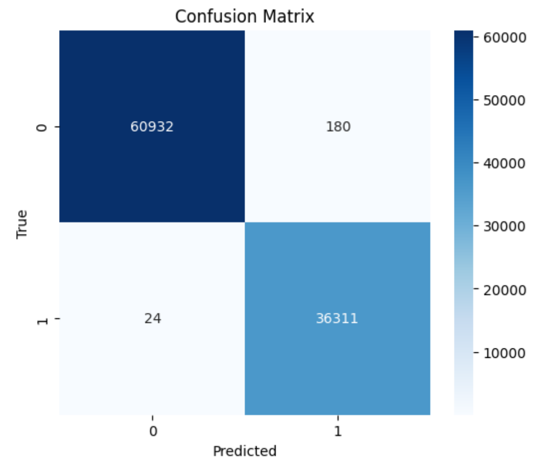


Fig. 31: First dataset confusion matrix

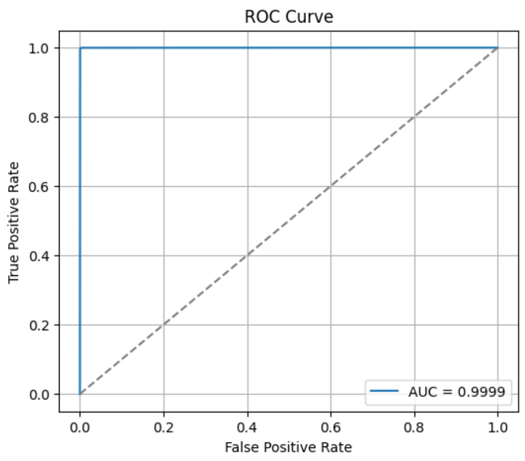


Fig. 32: First dataset ROC curve

d) **DNN**

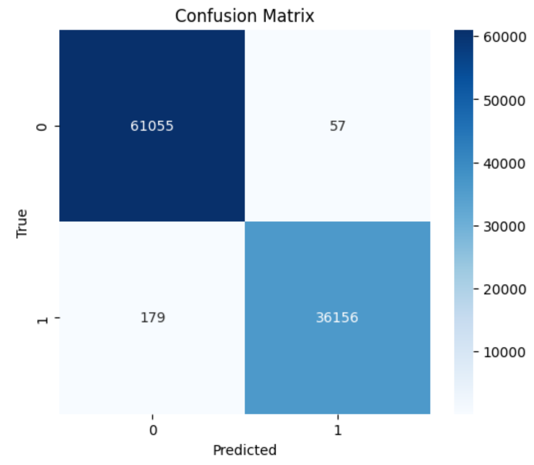


Fig. 33: First dataset confusion matrix

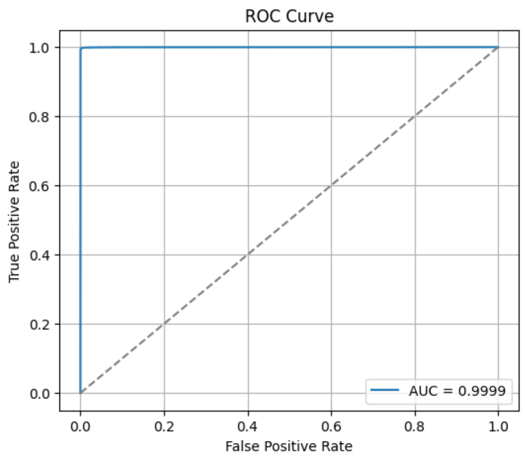


Fig. 34: First dataset ROC curve

e) **RoBERTa**

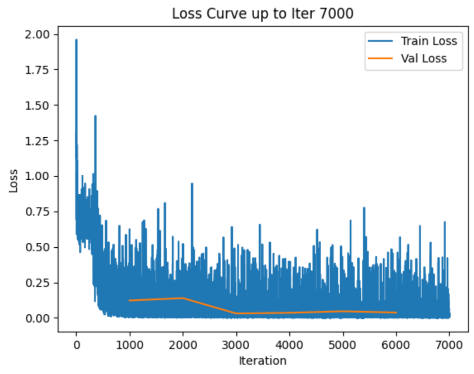


Fig. 35: Third dataset ROC curve

f) **ELECTRA**

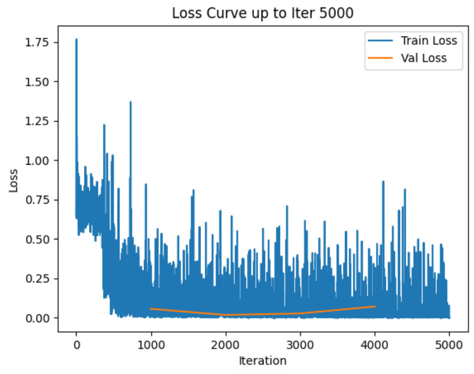


Fig. 36: Loss Curve for ELECTRA

The following results are from the second dataset.

a) **CNN**

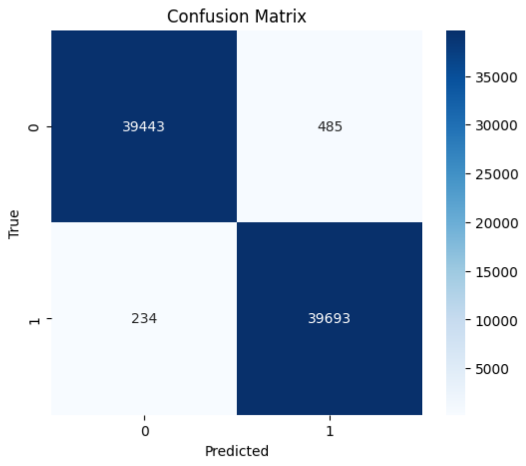


Fig. 37: Second dataset confusion matrix

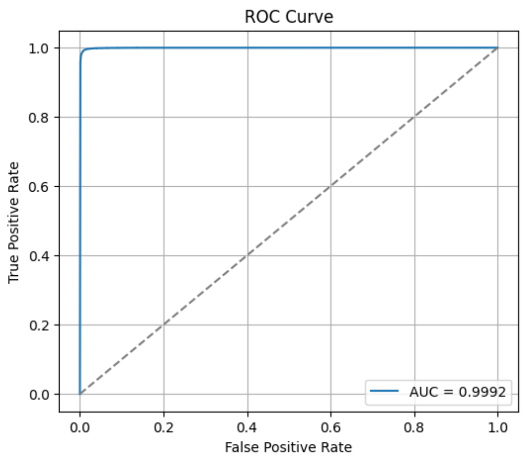


Fig. 38: Second dataset ROC curve

b) **BiLSTM**

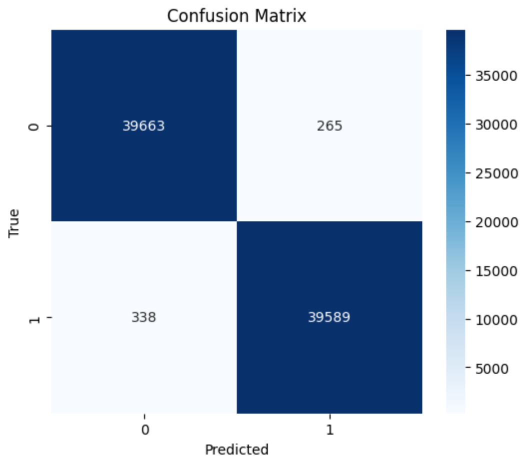


Fig. 39: Second dataset confusion matrix

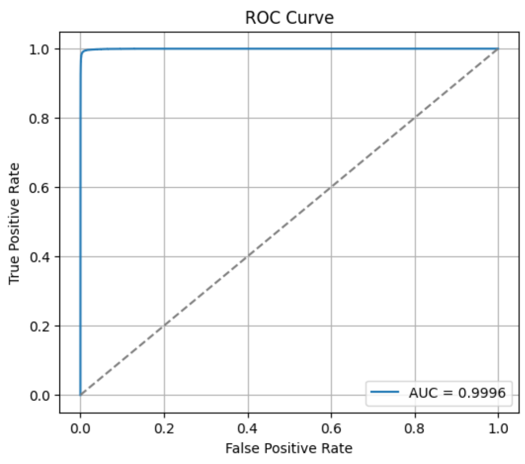


Fig. 40: Second dataset ROC curve

c) **GRU**

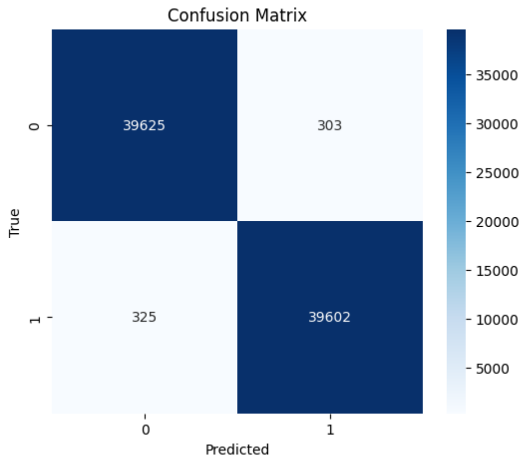


Fig. 41: Second dataset confusion matrix

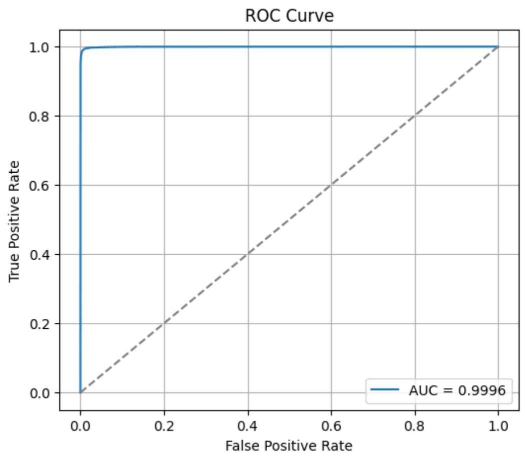


Fig. 42: Second dataset ROC curve

d) **DNN**

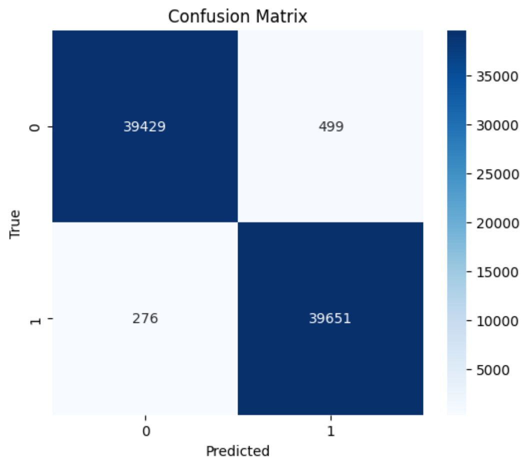


Fig. 43: Second dataset confusion matrix

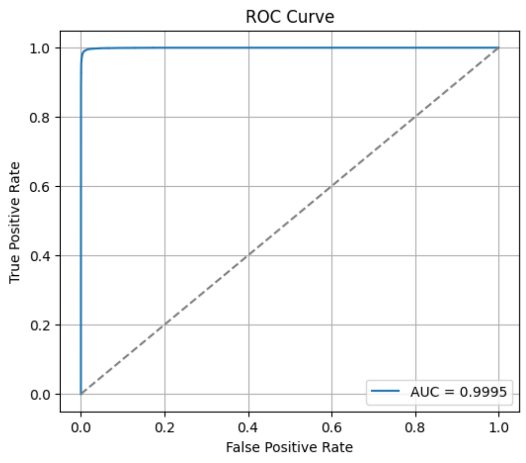


Fig. 44: Second dataset ROC curve

e) **RoBERTa**

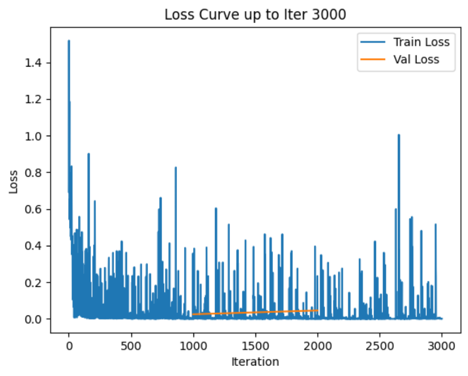


Fig. 45: Loss Curve for RoBERTa

f) **ELECTRA**

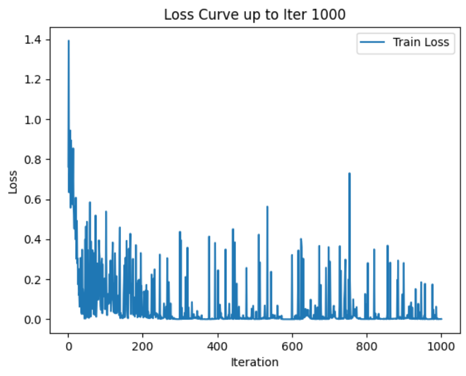


Fig. 46: Loss Curve for ELECTRA

The following results are from the third dataset.

a) **CNN**

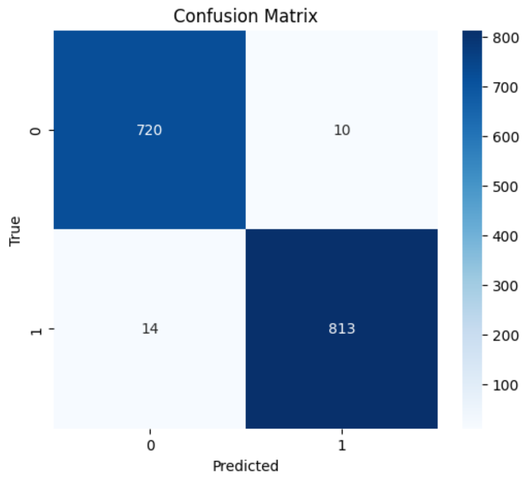


Fig. 47: Third dataset confusion matrix

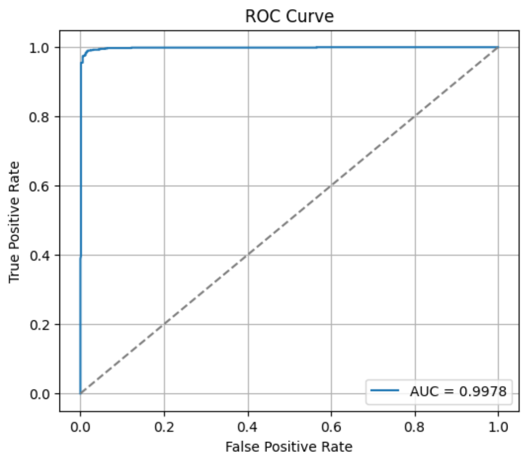


Fig. 48: Third dataset ROC curve

b) **BiLSTM**

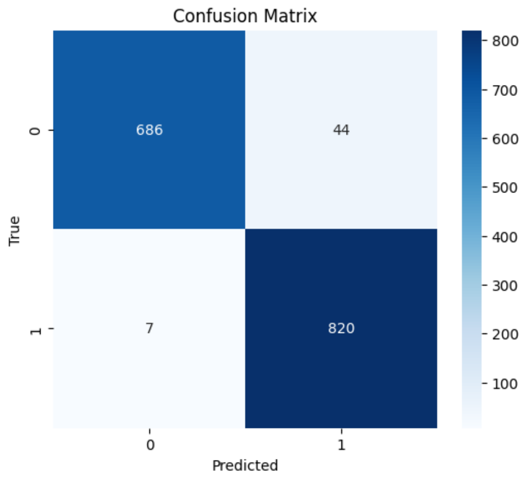


Fig. 49: Third dataset confusion matrix

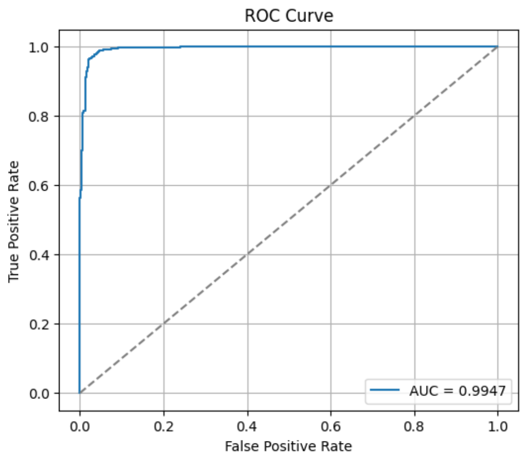


Fig. 50: Third dataset ROC curve

c) **GRU**

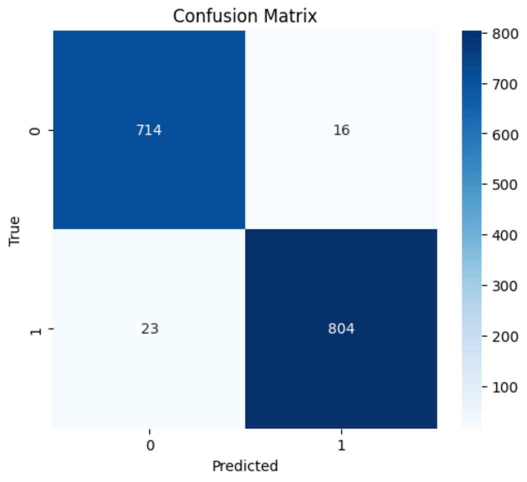


Fig. 51: Third dataset confusion matrix

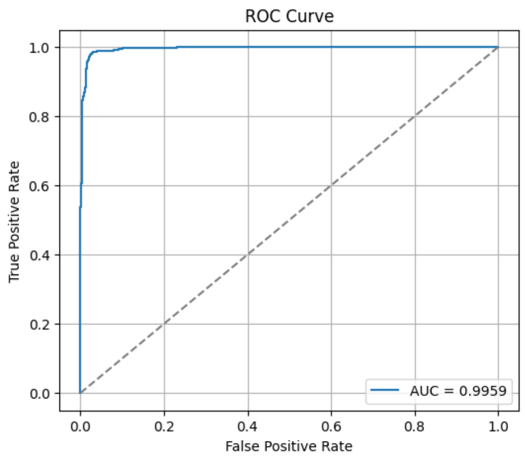


Fig. 52: Third dataset ROC curve

d) **DNN**

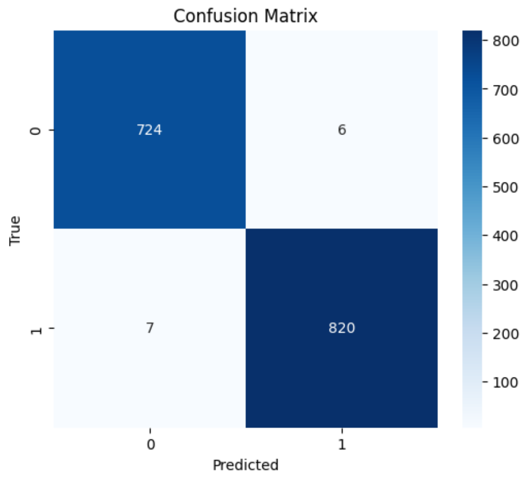


Fig. 53: Third dataset confusion matrix

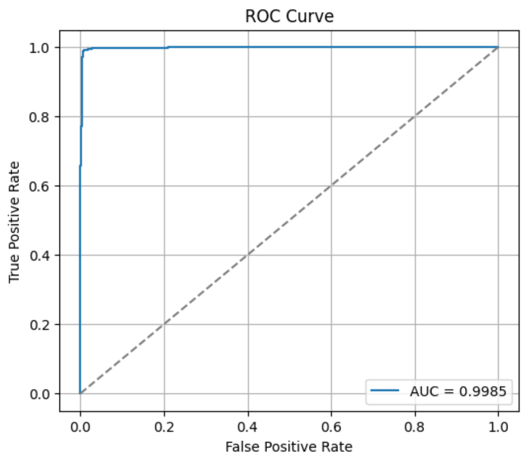


Fig. 54: Third dataset ROC curve

e) **AraELECTRA**

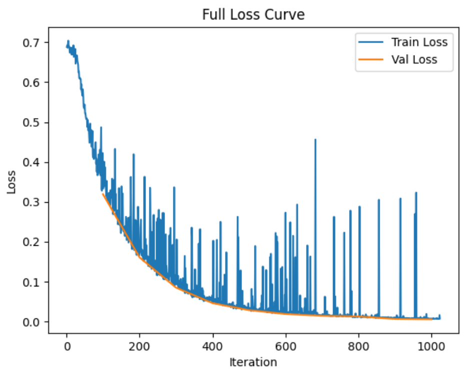


Fig. 55: Loss Curve for AraELECTRA

V. Conclusion

It is crucial for institutes and other entities to be able to detect AI in text as it helps them keep originality in their work and maintain their integrity. Machine learning is a proven great approach for text classification. For the first dataset, every model except NB have excelled in classifying if text is AI generated or not, with an accuracy of 99%. Regarding deep learning models, all models exceeded 99%. This is present for all three datasets For the second dataset, the best model was the passive aggressive classifier with an accuracy of 98.7%. As for the third dataset, Logistic Regression and Passive Aggressive were the best with an accuracy of 99%. Deep learning is becoming a very popular way for classification in many different fields making it very useful for humanity. Utilizing deep learning algorithms is an expanding field that will only keep on improving as time progresses.

References

[1] M. A. Flitcroft, S. A. Sheriff, N. Wolfrath, R. Maddula, L. McConnell, Y. Xing, K. L. Haines, S. L. Wong, and A. N. Kothari, "Performance of artificial intelligence content detectors using human and artificial intelligence-generated scientific writing," *Annals of Surgical Oncology*, vol. 31, no. 10, pp. 6387–6393, Oct 2024. [Online]. Available: https://doi.org/10.1245/s10434-024-15549-6

[2] D. Weber-Wulff, A. Anohina-Naumeca, S. Bjelobaba, T. Foltýnek, J. Guerrero-Dib, O. Popoola, P. Šigut, and L. Waddington, "Testing of detection tools for ai-generated text," *International Journal for Educational Integrity*, vol. 19, no. 1, p. 26, Dec 2023. [Online]. Available: https://doi.org/10.1007/s40979-023-00146-z

[3] A. M. Elkhatat, K. Elsaid, and S. Almeer, "Evaluating the efficacy of ai content detection tools in differentiating between human and ai-generated text," *International*

Journal for Educational Integrity, vol. 19, no. 1, p. 17, Sep 2023. [Online]. Available: https://doi.org/10.1007/s40979-023-00140-5

[4] H. Alshammari and K. Elleithy, "Toward robust arabic ai-generated text detection: Tackling diacritics challenges," *Information*, vol. 15, no. 7, 2024. [Online]. Available: https://www.mdpi.com/2078-2489/15/7/419

[5] A. Boutadjine, F. Harrag, and K. Shaalan, "Human vs. machine: A comparative study on the detection of ai-generated content," *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, vol. 24, no. 2, Feb. 2025. [Online]. Available: https://doi.org/10.1145/3708889

[6] R. Ardeshirifar, "Comparing hand-crafted and deep learning approaches for detecting ai-generated text: performance, generalization, and linguistic insights," *AI and Ethics*, Apr 2025. [Online]. Available: https://doi.org/10.1007/s43681-025-00699-4

[7] I. Katib, F. Y. Assiri, H. A. Abdushkour, D. Hamed, and M. Ragab, "Differentiating chat generative pretrained transformer from humans: Detecting chatgpt-generated text and human text using machine learning," *Mathematics*, vol. 11, no. 15, 2023. [Online]. Available: https://www.mdpi.com/2227-7390/11/15/3400

[8] K. Schaaff, T. Schlippe, and L. Mindner, "Classification of human- and ai-generated texts for different languages and domains," *International Journal of Speech Technology*, vol. 27, no. 4, pp. 935–956, Dec 2024. [Online]. Available: https://doi.org/10.1007/s10772-024-10143-3

[9] V. H. Tran, Y. Sebastian, A. Karim, and S. Azam, "Distinguishing human journalists from artificial storytellers through stylistic fingerprints," 2024. [Online]. Available: https://doi.org/10.3390/computers13120328

[10] H. Alshammari, A. ElSayed, and K. Elleithy, "Ai-generated text detector for arabic language using encoder-based transformer architecture," *Big Data and Cognitive Computing*, vol. 8, p. 32, 03 2024.

[11] H. Alharthi, "Investigation into the identification of ai-generated short dialectal arabic texts," *IEEE Access*, vol. PP, pp. 1–1, 01 2025.

[12] A. Al Bataineh, R. Sickler, K. Kurcz, and K. Pedersen, "Ai-generated vs. human text: Introducing a new dataset for benchmarking and analysis," *IEEE Transactions on Artificial Intelligence*, vol. PP, pp. 1–11, 01 2025.

[13] N. Krawczyk, B. Probierz, and J. Kozak, "Towards ai-generated essay classification using numerical text representation," *Applied Sciences*, vol. 14, no. 21, 2024. [Online]. Available: https://www.mdpi.com/2076-3417/14/21/9795

[14] C. Mordini, A. R. Vasquez, C. Zhang, M. Malinowski, D. Kienzler, K. Mehta, and J. Home, "Enhanced entangling gates and scalable multi-zone operations in surface traps with integrated photonics," in *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*, 2022, pp. 769–770.

[15] M. Perkins, J. Roe, B. H. Vu, D. Postma, D. Hickerson, J. McGaughran, and H. Q. Khuat, "Simple techniques to bypass genai text detectors: implications for inclusive education," *International Journal of Educational Technology in Higher Education*, vol. 21, no. 1, p. 53, September 2024. [Online]. Available: https://doi.org/10.1186/s41239-024-00487-w

[16] M. A. Wani, M. ElAffendi, and K. A. Shakil, "Ai-generated spam review detection framework with deep learning algorithms and natural language processing," *Computers*, vol. 13, no. 10, 2024. [Online]. Available: https://www.mdpi.com/2073-431X/13/10/264

[17] J. Blake, A. S. M. Miah, K. Kredens, and J. Shin, "Detection of ai-generated texts: A bi-lstm and attention-based approach," *IEEE Access*, vol. PP, pp. 1–1, 04 2025.